Detecting anomalies in multivariate time series from automotive systems

A Thesis submitted for the degree of Doctor of Philosophy

by

Andreas Theissler

Brunel University
London, United Kingdom
School of Engineering & Design

December 9, 2013

Abstract

In the automotive industry test drives are conducted during the development of new vehicle models or as a part of quality assurance for series vehicles. During the test drives, data is recorded for the use of fault analysis resulting in millions of data points. Since multiple vehicles are tested in parallel, the amount of data that is to be analysed is tremendous. Hence, manually analysing each recording is not feasible. Furthermore the complexity of vehicles is ever-increasing leading to an increase of the data volume and complexity of the recordings. Only by effective means of analysing the recordings, one can make sure that the effort put in the conducting of test drives pays off. Consequently, effective means of test drive analysis can become a competitive advantage.

This Thesis researches ways to detect unknown or unmodelled faults in recordings from test drives with the following two aims: (1) in a data base of recordings, the expert shall be pointed to potential errors by reporting anomalies, and (2) the time required for the manual analysis of one recording shall be shortened.

The idea to achieve the first aim is to learn the normal behaviour from a training set of recordings and then to autonomously detect anomalies. The one-class classifier "support vector data description" (SVDD) is identified to be most suitable, though it suffers from the need to specify parameters beforehand. One main contribution of this Thesis is a new autonomous parameter tuning approach, making SVDD applicable to the problem at hand. Another vital contribution is a novel approach enhancing SVDD to work with multivariate time series. The outcome is the classifier "SVDDsubseq" that is directly applicable to test drive data, without the need for expert knowledge to configure or tune the classifier. The second aim is achieved by adapting visual data mining techniques to make the manual analysis of test drives more efficient. The methods of "parallel coordinates" and "scatter plot matrices" are enhanced by sophisticated filter and query operations, combined with a query tool that allows to graphically formulate search patterns.

As a combination of the autonomous classifier "SVDDsubseq" and user-driven visual data mining techniques, a novel, data-driven, semi-autonomous approach to detect

unmodelled faults in recordings from test drives is proposed and successfully validated on recordings from test drives. The methodologies in this Thesis can be used as a guideline when setting up an anomaly detection system for own vehicle data.

Acknowledgements

I have heard people describing a PhD study as a long journey of ups and downs. Since Jule Verne's "Around the World in Eighty Days", we know how far you can travel within just 80 days. So in terms of this metaphor I would describe writing a PhD Thesis as going around the world several times. By all means, it takes a prominent place in one's life over a period of several years. Especially in the final year, everything revolved around writing up the Thesis. While I always believed that I would reach the end, I was not always sure about how and when.

I would like to express my gratitude to a number of people and institutions that supported me throughout my study in one way or the other.

I want to thank Brunel University for making it possible to write my PhD in industry in Germany. In particular, utmost gratitude to my supervisor Dr Ian Dear who supported me throughout the entire time of the PhD. He always managed to find time slots in his busy schedule where we had various extensive meetings that were precedent-setting for the study. I would also like to thank my second supervisor Prof John Stonham for insightful discussions and questions from an additional point of view and I wish to thank Prof. Dr. Dominik Schoop for insightful discussions, especially in the beginning of the PhD.

This work was conducted in a research project run by IT-Designers GmbH and STZ Softwaretechnik in Esslingen, Germany. I am most grateful for their funding of this research project, in particular to Prof. Dr. rer. nat. Joachim Goll for creating an environment that gave me enough free space to follow my ideas, for his comments and questions on my work, and for keeping most of the day-to-day work away from me during the time of writing up the Thesis. Furthermore, I want to thank the participants of the biannual PhD workshop, in particular the organiser Prof. Dr. rer. nat. Manfred Dausmann.

I wish to thank my colleagues at IT-Designers GmbH and STZ Softwaretechnik, especially the entire development team of "Tedradis", a system for recording and analysis of test drives. This was the origin for the research presented in this work and some of the

implementation results will be or have already been integrated in this product. I am thrilled to see part of the research work being transferred from research to industrial usage. Furthermore I want to thank Angelika, Gudrun, Maria, and Petra.

In the research project a high number of students and various colleagues were employed. Normally due to time constraints in a PhD, a rapid prototype to validate the ideas is developed. The high number of students and colleagues allowed the development of a fully functioning system.

Sincere thanks to Daniel Hommel, Stephan Pressler, Steffen Brauns, Falk Kleehammer, Matthias Kohles, Daniel Weber, Alexander Kraft, Paul Sprecher, Mikhail Orleansky, Oleksandr Pavlichenko, Roland van der Schoot, Stephan Frey, Marc Gerhard, Fikret Kaplan, Jens Ehlert, Moritz Bahr, and in particular to Gunnar Niess who has worked with me over a period of more than two years.

Thank you to the proof readers who helped to find those weaknesses in the text that the author himself is blind to: Micky Sung, Jens Ehlert, Gunnar Niess, Ralf Schmidgall, Peter Schlumberger, and Steffen Wahl.

Many thanks to the various test drivers, in particular Jens Ehlert, Bernd Theissler, and Gunnar Niess, and to the car repair shop Kfz & Zweiradtechnik Schwarzer for valuable hints on fault injection.

A supportive environment is important to get through a PhD, so my sincere thanks for the support of my friends, parents, brothers, and parents-in-law. A special "thank you" to my former colleague Niko, who encouraged me to go to university after my apprenticeship by saying the following sentence in his German with Croatian accent again and again: "Boy, go back to school when you are still young. You'll have enough time to work when you are older."

Finally, my utmost gratitude to those people who suffered most from me doing my PhD: my wife Sandra, my daughter Emily, and my son Julian ("Vielen Dank für die Unterstützung!"). They say, if you watch children growing up, you'll notice how you are getting older. Doing your doctorate you'll also notice, how long it takes to get a PhD.

Thank you!

Contents

Glossary

pdf
: Probability density function. 93–95, 97, 100, 101

SVDDSUBSEQ
: The classification approach for multivariate time series proposed in this Thesis. 175, 178, 179, 182–184, 187, 193, 195, 197, 199, 201, 205, 206, 238, 242, 245, 246, 252

anomaly
: A deviation in the behaviour of a univariate time series or in the relationship of multiple univariate time series from expected behaviour. 37

AR
: Autoregressive model. A model that can be used to predict future values of a time series or to generate time series. 183

ARMA
: Autoregressive moving average model. A model that can be used to predict future values of a time series or to generate time series. 54, 183

AUTOSAR
: Automotive Open System Architecture. A standardised layered automotive software architecture. 12

CAN
: Controller Area Network. Network technology used in vehicles to interconnect electronic control units. 7

ECU
: Electronic control unit. ECUs read data measured by sensors and calculate control values based on the sensor input. The control values are then transmitted to actuators. 1

error
: The discrepancy between a computed, observed or measured value or condition,

and the true, specified or theoretically correct value or condition. An error is viewed as an anomaly in this work. 30

fault
An abnormal condition that can cause an element or an item to fail. 30

HiL
Hardware-in-the-loop. Test rigs where an ECU is tested in a simulated environment.. 3

k-NN
k-nearest neighbours. An instance-based classifier capable of learning non-linear decision boundaries. 53, 110, 127

LIN
Local interconnect network. Cost-effective master-slave network technology used in vehicles. 7

LOF
Local outlier factor. An instance-based technique that assigns LOF values to individual instances based on distances and densities. 127

MA
Moving average models. A model that can be used to predict future values of a time series or to generate time series. 183

MOST
Media Oriented Systems Transport. High-speed network technology used for infotainment systems in vehicles. 7

multivariate time series
Consists of $M > 1$ univariate time series, i.e. for every time point t_i there exist M data points d_i. 38

OSEK
"Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug". Real-time operating system that runs on electronic control units. 7

precision
The percentage of true anomalies in the result set of all instances classified as anomaly. 92

RBF
Radial basis function. Kernel used in support vector machines, also referred to as Gaussian kernel. 144

recording
Discrete time-stamped signal values recorded during vehicle tests. 38

SAX
Symbolic aggregate approximation. Symbolic representation of a time series. 67, 88

SiL
Software-in-the-loop. Test environment where an ECU's software is tested in a simulated environment. 3

SVDD
Support vector data description. A one-class classifier based on support vector machines. 133

univariate time series
Finite sequence of data points ordered by time and denoted by X_T. 38

Symbols

$P(\omega_c)$	Prior probability of class ω_c. 93
W	Length of subsequence in data points. 176
$\mathbb{I}$	Data set in input space. 89
$\mathbb{I}_v$	One instance in input space. 90
$\mathcal{A}$	Training data set in feature space. 89, 100, 102–104, 106, 107, 110, 112, 113, 120, 121, 151, 178, 179
$\mathcal{B}$	Test data set in feature space. 89, 121, 179
$\mathcal{C}$	Unseen data set in feature space. 89, 91
$\mathcal{C}_v$	Instance in unseen data set in feature space. 91, 93, 102–104, 107, 110, 113, 120
$\mathcal{F}$	Data set in feature space. 89, 98, 102, 108, 111, 151, 154, 158
$\mathcal{F}_v$	One instance in feature space, i.e. one feature vector. 90
$\mathcal{N}(x_{test})$	Set of nearest neighbours. 128, 130, 132, 133
f	A feature. 90, 94, 95, 109, 111
ω_a	Abnormal class. 91, 92, 94, 97, 99, 103–105, 108, 109, 111, 120, 164, 168–172
ω_c	Class c. 91, 93, 99, 110
ω_n	Normal class. 91, 92, 94, 97, 99, 103–105, 108, 109, 111, 120, 121, 164, 168–172, 175
ϕ	Mapping function from input space to feature space. 89
$\phi(\mathcal{F})$	Transformed feature space. 151, 154, 157, 158
$d(\mathcal{F})$	Decision function. 102
e_{Bayes}	Bayesian error. 98–100, 124
e_{ω_n}	Error on the normal class. 151, 152, 154, 157, 158, 170, 172
$p(f_p\|\omega_n)$	Probability density function of normal class. 121
FN	False negatives, i.e. falsely detected anomalies. 91
FN/h	Number of normal subsequences incorrectly classified as abnormal in one hour of data. 215, 223, 225–227, 229, 231, 235, 236
FP	False positives, i.e. number of anomalies falsely classified as normal. 91

TN	True negatives, i.e. number of correctly detected anomalies. 91
TNR	True negative rate, i.e. percentage of correctly detected anomalies. 92
TP	True positives, i.e. normal instances correctly classified as normal. 91
TPR	True positive rate, i.e. percentage of normal instances correctly classified as normal. 92

Chapter 1

Introduction

This chapter gives an introduction into vehicle electronics and identifies the shortcomings in the current process of analysing recordings from test drives in the automotive industry. Based on this, the scope and the aims of the Thesis are identified.

125 **years** after the invention of the automobile in 1886, vehicles have turned from mechanical machines into highly complex products dominated by software and electronics. This applies in equal measure to cars, trucks, busses, and agricultural engines (Schlingmann, 2008). Nowadays, the gross of innovations in the automotive industry is achieved by means of software and electronics. Following an expected annual growth between 6 percent (Dannenberg and Burgard, 2007) and 9 percent (Mayer, 2010d), the market for vehicle electronics is estimated to have a yearly volume of approximately 200 billion British Pounds by the year 2015 (Mayer, 2010d).

Modern vehicles have 40 to 80 electronic control units (ECUs) communicating over several bus systems. An example of an ECU would be the unit controlling the vehicle's engine. The ECUs read data measured by sensors and calculate control values based on the sensor input. The control values are then transmitted to actuators. The

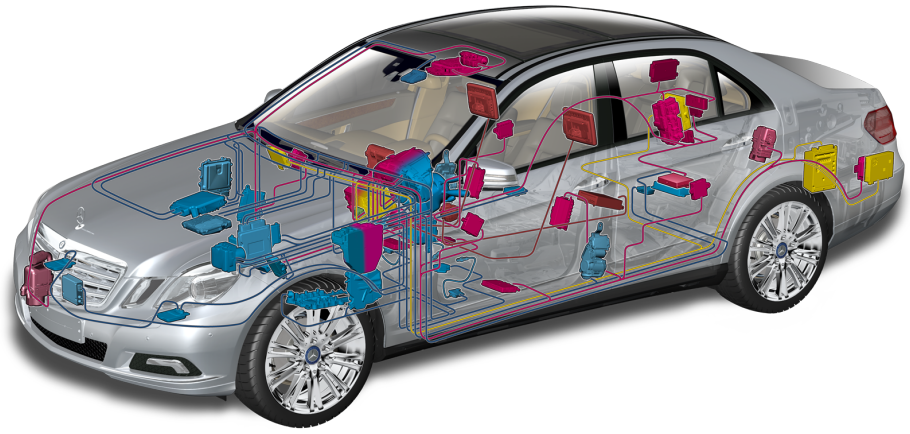


Figure 1.1: Electric and electronic components and in-vehicle network in a premium class car (taken with permission from (Schmidgall, 2011))

sensors' and actuators' values are being transmitted over the bus systems and received by further ECUs (Marscholik and Subke, 2008). This results in a highly complex, heterogeneous network of software and hardware subsystems delivered by a variety of suppliers. As a consequence, there is a high potential for faults either caused by one individual subsystem or by the integration of those subsystems into the vehicle. An example of how ECUs exchange signals is the determination of the vehicle's velocity: An ECU reads the raw data from the speed sensors in the wheels and calculates the individual wheels' velocities. The vehicle's velocity in turn is calculated based on the combination of the wheel speed values and is then transmitted to further ECUs together with the velocities of the wheels. The electric and electronic components together with the interconnecting networks inside a current premium class car are shown in Figure 1.1.

This chapter surveys the measures of quality assurance for electric and electronic components of vehicles and identifies the shortcomings, where this Thesis focuses on test drives. Starting with an overview of the state of the art in vehicle electronics, the driving factors of the automotive industry are identified and forecasts about the development of vehicle electronics are stated. Conclusively the implications on the analysis of recordings from test drives are deduced.

The underlying question that motivated this Thesis is:

How can we cope with the soaring data volume and complexity of recordings from test drives caused by the ever-increasing complexity of vehicles?

1.1 Assuring quality in vehicle electronics

In (Lamberg, 2006) the test phases for electric and electronic components in vehicles are sketched, starting with the testing of an executable function model, i.e. the function itself is not implemented. The next steps are software tests, testing the implemented function code, followed by hardware-in-the-loop (HiL) tests of individual ECUs. Within the system test, an ECU is tested together with its immediate environment, like further ECUs or sensors. The last test phase is the integration test, where all ECUs are integrated in the vehicle and the full vehicle is tested in the production plant and on the road.

1.1.1 Quality assurance in the laboratory

The testing of a software component starts with software-in-the-loop simulation (SiL), where the function is tested in a completely simulated environment. The software component is not running on its target platform, but typically on a PC. At a later stage, the software component is deployed on the ECU and tested by hardware-in-the-loop (HiL) simulation, where the ECU is real and its environment, like sensors and actuators, can be virtual or real (Schäuffele and Zurawka, 2005).

The entire test process validates that a function works as specified, that implausible input values are handled properly, and that the list of specified vehicle faults are detected by an ECU and the corresponding diagnostic trouble codes are stored. In none but very simple cases is exhaustive testing feasible due to time and budget constraints. One rather constrains himself to that subset of test cases, expected to have the highest probability of detecting the most errors (Myers, 2004).

As opposed to testing, which by definition can never be used to prove the absence of faults (Myers, 2004), the proof of correctness can be achieved by formal verification. The application of formal verification to vehicle electronics has been reported, but is currently at a research stage. In (Sander et al., 2009) it was shown for the verification of a LIN communication controller. In (Endres et al., 2010), the authors could prove that the scheduling of a prototypical automatic emergency call system based on FlexRay is correct.

While the author views the approaches of formal verification as promising for the verification of critical parts, as shown for the communication controller, the author strongly believes that the full verification of an entire vehicle will remain infeasible. This belief is backed by the observation that on the one hand the complexity of vehicles increases, on the other hand the time for development and testing decreases.

1.1.2 Quality assurance by test drives

Even though various test phases are conducted for each ECU and for each vehicle function (Lamberg, 2006), the integration of all ECUs inside a vehicle, with real sensors, actuators and the real in-vehicle network, is challenging – often unexpected problems occur. Hence, conducting test drives is inevitable. No other test phase works under more realistic conditions. Faults that occurred during test drives but remain undetected are likely to occur in the field as well, where they become obvious to customers.

The importance of test drives is stated by a vehicle manufacturer in (Krämer et al., 2009): before one of its premium class models came to market in 2009, 21 million miles of test drives were conducted and based on these tests the degree of maturity was deduced.

In order to be able to locate faults or to evaluate the behaviour of vehicle subsystems, the communication on the in-vehicle network and in some cases internal variables of ECUs are being recorded by data acquisition systems (automotive data loggers) during the test drives. In addition, for specific tests external sensors are incorporated.

During fault analysis the recordings allow the reconstruction of the situation that the vehicle was in, e.g. abrupt steering manoeuvres, the vehicle's velocity, its yaw rate or the battery voltage can be determined from the data. This kind of recordings are conducted by manufacturers with prototype vehicles during test drives before start of production or with series vehicles as part of the end of line tests in the process of quality assurance.

The following approaches are currently followed by vehicle manufacturers in order to detect errors occurring in test drives:

1. Illegal constellations of the signals are pre-configured and the measured data is monitored by the data acquisition system.

2. During or after the test drive, diagnostic testers are used to read out the ECUs' diagnostic trouble codes (Marscholik and Subke, 2008) to check if the ECUs' fault detection mechanisms detected a fault.

3. After the test drive, the recordings are searched for known, pre-configured fault patterns.

4. A test driver manually flags suspicious or erroneous vehicle behaviour during the test drive.

5. Rarely, a small fraction of the recordings is manually investigated by an expert after the test drive by random inspection.

Contemplating the measures taken during test drives, shows that the first three points rely on pre-configured knowledge. As a consequence, only those faults are detected that were modelled during the definition of the test strategy. The fourth and fifth point solely rely on the experience of human beings. However, none of the listed measures explicitly handles unmodelled faults.

The different ways currently used for analysing test drive data, subdivided into the detection of known/modelled faults and unknown/unmodelled faults, are shown in Figure 1.2.

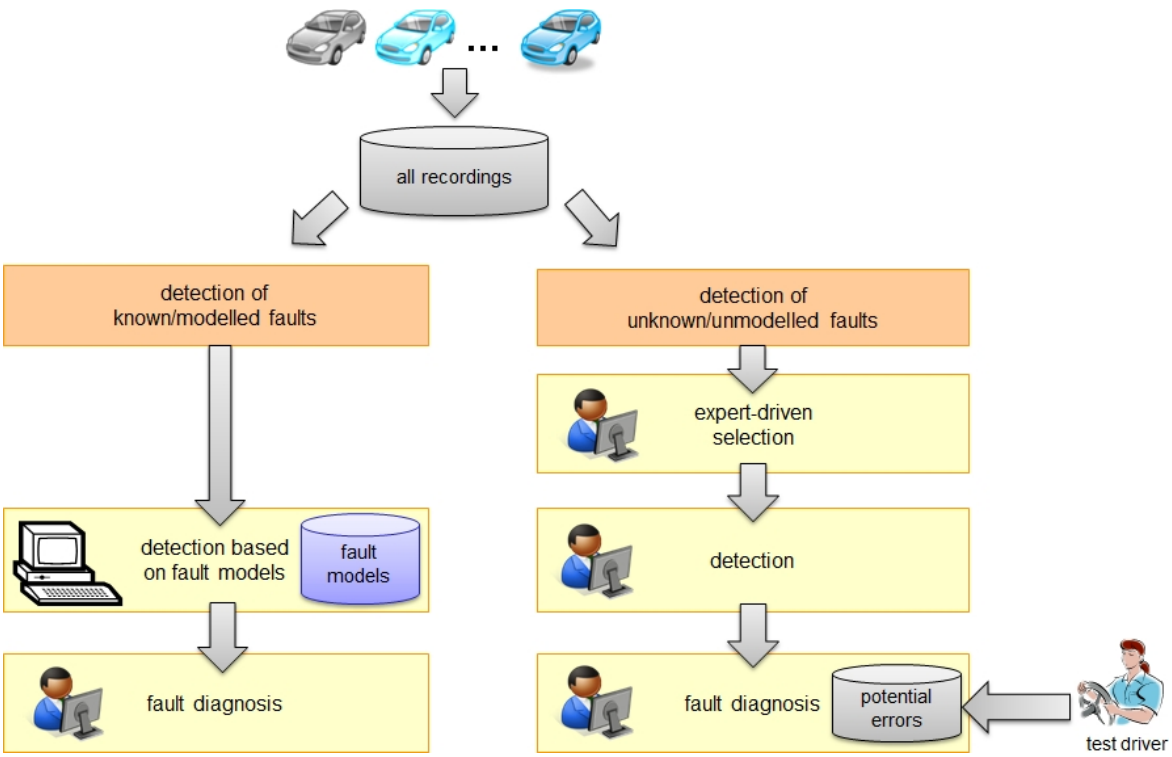


Figure 1.2: Current process of test drive analysis.

Known and modelled faults are detected based on fault models either by data loggers during test drives or during post-processing of the data. Unmodelled faults are detected by analysing test driver comments or investigating a small portion of the recordings, selected either randomly or based on expert experience.

Figure 1.3 shows the improvement with the approach proposed in this Thesis. Instead of data selection and detection of potential errors based on expert-knowledge, these two steps are fully automated, saving the expert valuable time.

Testing vehicles on the road is a very cost-intensive process, because it involves a high fraction of manual work: The test drives are conducted by test drivers or test engineers. As opposed to testing techniques running on computers or test rigs, the test equipment itself involves real vehicles – often hand-crafted, precious prototypes. Hence, the author views the recordings of test drives themselves as very precious.

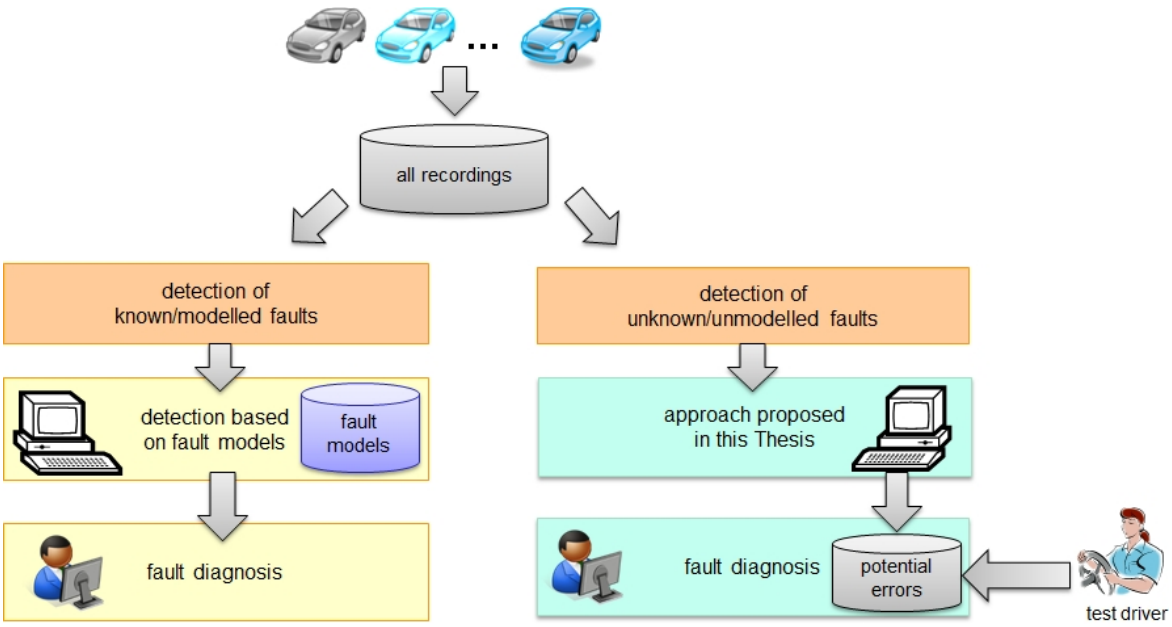


Figure 1.3: Process of test drive analysis with the approach proposed in this Thesis automating the selection of subsets of data and the detection of potential faults.

1.2 Vehicle electronics: State of the art

In this section the state of the art in vehicle electronics is introduced. Vehicle electronics comprises sensors, actuators, ECUs, the in-vehicle network connecting the components, and software components running on ECUs.

The software on ECUs runs inside pre-scheduled tasks on real-time operating systems implementing the OSEK standard (Schäuffele and Zurawka, 2005). The ECUs are interconnected by an in-vehicle network, which can be viewed as a heterogeneous distributed system with a variety of different network technologies and hardware platforms. The different subnetworks are connected by gateways. Figure 1.4 shows a simplified example of an in-vehicle network with CAN, LIN, MOST, and FlexRay subnetworks – currently the four most widely used standards. At the moment not all manufacturers have introduced all of these standards.

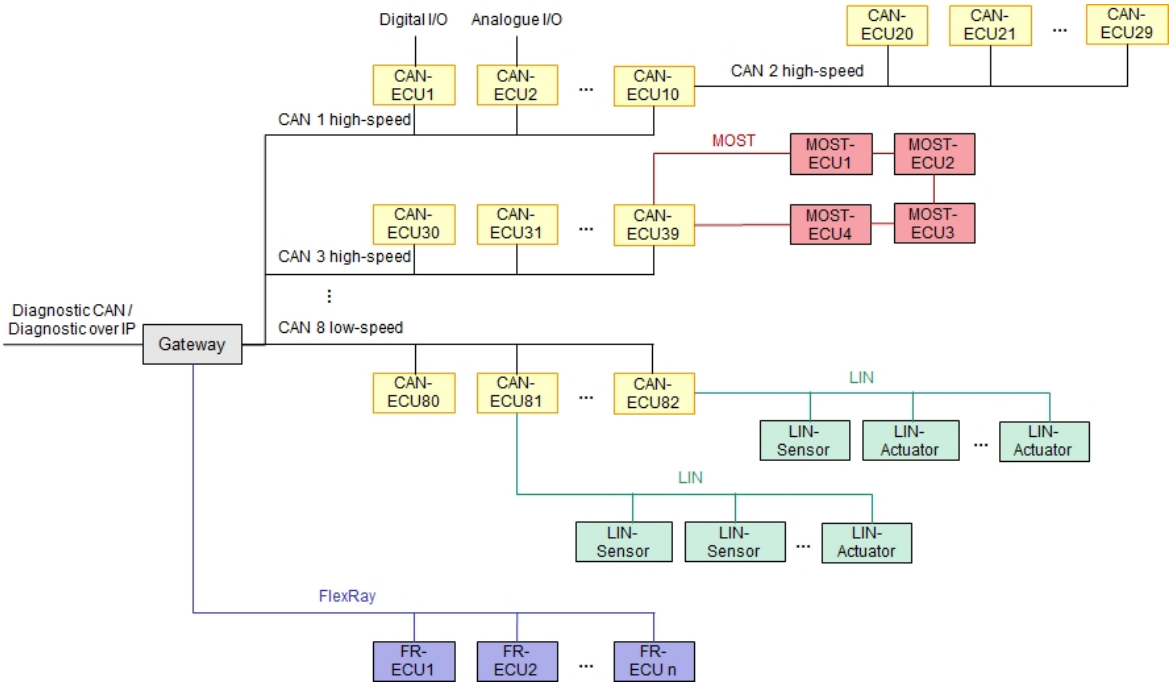


Figure 1.4: A simplified example of an in-vehicle network with CAN, LIN, MOST and FlexRay bus systems

At the time of writing, a vehicle network is dominated by CAN (Controller Area Network) networks. CAN (Mayer, 2010a) is a non-deterministic network with a theoretical maximum bandwidth of 1 MBit/s. In vehicles CAN is typically operated at 500 kBit/s for high speed CAN (ISO 11898-2) and 125 kBit/s for low speed CAN (ISO 11898-3). Bus access is controlled based on message priorities (Mayer, 2010a). Between two and ten CAN subnets are currently present in modern vehicles. Applications can be found throughout the entire vehicle, e.g. for powertrain and chassis functions.

As a cost-effective substitution for direct wiring of sensors and actuators, LIN (Local Interconnect Network) (Mayer, 2010b) is used. LIN is a cost-effective master-slave network technology for non-critical data communication with a maximum bandwidth of 20 kBit/s. Applications can be found in a vehicle's convenience area, e.g. mirror adjustment or electric window openers.

For infotainment applications with high-bandwidth demands, manufacturers use MOST (Media Oriented Systems Transport) (Grezemba, 2011), which is a high-bandwidth network operating in a ring topology in master-slave mode. There are standards ranging from 25 MBit/s up to 150 MBit/s. MOST is for example used for the streaming of audio and video signals (Mayer, 2010d).

In order to overcome some of the weaknesses of the predominantely used CAN, the FlexRay standard was developed. FlexRay (Mayer, 2010c) is a deterministic and fault-tolerant network operating at a bandwidth of 10 MBit/s at each of two channels. The communication follows a pre-defined communication schedule using Time Division Multiple Access (TDMA) (Mayer, 2010c). The first application of FlexRay in a series vehicle was the dynamic adaption of a vehicle's dampers to the current driving situation (Jautze et al., 2008).

In addition, consumer electronics like portable music players are connected to the vehicle using Bluetooth or USB. Also, Wireless LAN is likely to be introduced in future vehicles allowing for car-2-car or car-2-infrastructure functionality (Grimm et al., 2007). To a minor extent, legacy or proprietary network technology can be found in subnets of current vehicles. Finally, in order to allow workshops to access the ECUs, there is one mandatory CAN bus connecting the in-vehicle network via a gateway to

a standardised socket, the so-called onboard-diagnostics (OBD) interface (Marscholik and Subke, 2008).

1.3 Vehicle electronics: Forecasts and implications

Based on the contemplation of current trends, forecasts are deduced and the implications on the data volume and complexity of recordings from test drives are discussed in this section.

The automotive industry is driven by a number of factors that will lead to technological advancements. The author believes that most technological advancements of vehicles will be observable as an increase in the complexity and variability of the in-vehicle network communication. In (Dannenberg and Burgard, 2007) it is stated, that 60% of the innovations in the automotive industry are driven by vehicle electronics. The identified key factors are given in this section.

A major challenge for the automotive industry is the reduction of the vehicle emissions. This challenge is addressed by advancements of combustion engines and by the introduction of vehicles with alternative drive trains, e.g. pure electric vehicles, fuel cell vehicles, and different hybrid concepts of combustion and electric engines. In order to keep development and production costs low, manufacturers are aiming to have more variable drive train concepts for the same vehicle platform (Wolfsried, 2009; Hackenberg, 2008). A vehicle will be available with different drive trains, e.g. using plug-in concepts. The author expects this to result in a higher complexity of the communication on the in-vehicle network, which is backed by the assessment given in (Weinmann et al., 2009).

Efforts to save energy also change the in-vehicle network from a static to a dynamic topology. While up to now all ECUs are permanently active, as long as the ignition is switched on, in the near future ECUs will be selectively switched to sleep mode in order to save energy. This means that the constellation of currently active network nodes will dynamically change. In (Schlinkheider, 2010) the definition of subsets of

ECUs that will be switched together is proposed. An example of 16 subsets shows the additional dimension of complexity that is introduced. While at the moment all ECUs are either switched on or off, with 16 subsets, the number of different, valid states is 2^{16} if distinguished between the states on and off. Additional states like standby-mode are also possible.

Furthermore, advancements in vehicle electronics are indirectly driven by legislation, e.g. by the introduction of an upper limit of acceptable vehicle emissions or by the demand of an electronic stability control (ESC) (Liebemann et al., 2004) for new vehicles in the future in the European Union and the USA (von Glasner and Micke, 2010).

The complexity of in-vehicle networks will further increase in order to satisfy the market's demands. Customers demand electronic safety functions like collision mitigation systems (Jones, 2001) or autonomous lane keeping. This kind of driver assistant functions combine data from various sensors (Pons et al., 2010). In other words, the data of one sensor is shared by different vehicle functions running on different ECUs, which means the data needs to be transmitted over the network. In addition, more advanced comfort functions have been developed, like e.g. autonomous parking. Current research addresses autonomous driving (Berger and Rumpe, 2012).

The number of vehicle functions implemented as software on the ECUs is in the region of 1000 in a premium class car (Wernicke, 2010). An example of intensive networking between ECUs is given by a vehicle manufacturer in (Wolff, 2009) for a premium class car in the year 2009. The adaptive cruise control system, a driver assistance function (von Glasner and Micke, 2010) monitoring the distance to the preceding vehicle and thereby controlling the vehicle's velocity, integrates the number of 26 ECUs. The trend of an increasing number of vehicle functions realised by the interconnection of several ECUs and sensors will continue, resulting in a further increase in complexity. The increase of the number of functions and ECUs in a car is shown in Figure 1.5 which was taken from (Dannenberg and Burgard, 2007). It is further shown that a growing number of innovations is not implemented as an isolated subsystem but rather by combining vehicle subsystems.

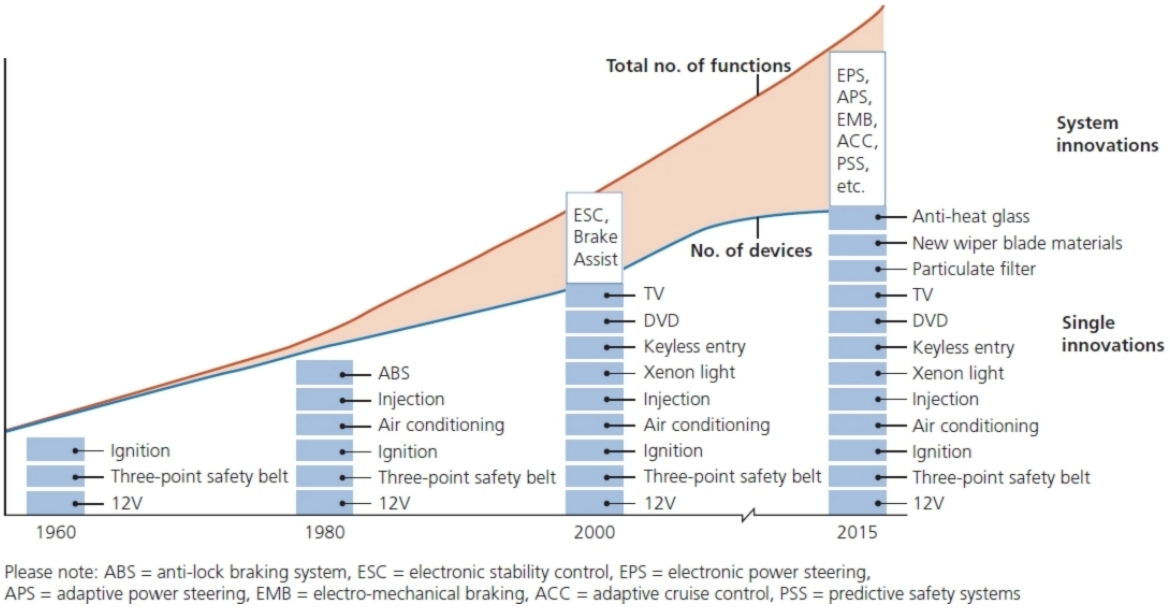


Figure 1.5: An excerpt of the increasing number of functions and ECUs taken from (Dannenberg and Burgard, 2007).

Finally, the in-vehicle network communication will be affected by advancements in vehicle electronics itself, like the introduction of the standard AUTOSAR (Automotive Open System Architecture) (Fuerst, 2010). AUTOSAR allows for platform-independent development of vehicle functions by introducing a layered architecture, comparable to virtual machines on PCs. At design stage, all software components communicate over a so called virtual function bus (VFB), only during deployment the decision is made on which of the ECUs a software component will run and thereby which of the signals will be exchanged via the network. Conclusively AUTOSAR allows for a more flexible distribution of functions among ECUs, which as a consequence is believed by the author to lead to a wider variety in the network communication.

Further changes can be observed as a result of advancements in the network technology, as can currently be seen by the introduction of the bus system FlexRay. With a bandwidth of 10 MBit/s, the data volume transmittable over FlexRay is higher than it is using the widely used CAN bus with a maximum bandwidth of 1 MBit/s. But more importantly, as opposed to the CAN bus, FlexRay is a deterministic bus system, allowing the use of the in-vehicle network for applications with real-time demands,

so e.g. closed-loop control can be distributed among several ECUs using FlexRay. Network technology with deterministic and fault-tolerant behaviour allows for the development of x-by-wire systems like steer-by-wire and brake-by-wire (Mayer, 2010c). At the time of writing, various manufacturers have started to use FlexRay in series vehicles. Additionally, a new standard for CAN is currently developed allowing for a higher bandwidth (CAN-FD) and Ethernet is introduced in current vehicles.

Conclusively the author states that the identified driving factors of the automotive industry will lead to an increasing complexity of vehicle electronics observable by an increasing communication effort on the in-vehicle network. Hence, in order to keep quality constant, a higher effort for quality assurance will be necessary.

1.4 Motivation and aims

Analysing the recordings of test drives manually is time consuming and exhausting. Each recording has to be contemplated and evaluated by a domain-expert. The amount of data resulting from each recording is huge. Depending on the configuration of the data acquisition system, thousands of signals with cycle times being in the region of milliseconds can be recorded over a time of several hours resulting in huge data volumes. So for example, recording 1000 signals with an assumed average cycle time of 50 ms over a period of one hour leads to $1000 * \frac{3600s}{0.05s} = 72$ million data points. Since multiple vehicles are being tested in parallel, the amount of data to be analysed is tremendous. Assuming a vehicle fleet of 50 vehicles, with each vehicle driven 10 hours a day, the data to be analysed is 36 billion data points per day.

Consequently, manually analysing each recording using current techniques is not feasible due to the huge amount of data. The expert should (1) be disburdened from analysing the entire recordings in great detail and (2) be supported during the manual analysis of one recording. A system supporting the expert by preventing her/him from contemplating irrelevant data and rather pointing the expert to the relevant parts would be very beneficial.

The benefit of such an approach shall be made clear by looking at an example of the currently used random inspection by an expert. Say, there are 1000 recordings of equal length and data volume. If it takes an expert one hour to properly analyse one recording and there is an available time budget of 20 hours, the expert will pick 20 recordings by random inspection and analyse them. In other words, most of the recordings will not be analysed, which is what happens in practise.

This Thesis researches ways of optimising the described process of analysing recordings from test drives with the following two aims:

Aim 1: in a data base of recordings from test drives point the expert to potential errors by reporting anomalies in the recordings

Aim 2: shorten the time required for the manual analysis of one recording

Achieving these aims would improve the detection rate of unmodelled faults, reduce the time required for fault detection and thereby allowing to conduct the analysis of suspicious data at a greater depth.

In Figure 1.6 the current and the new approach are compared. From the data base of recordings D_{all}, k fixed-size fractions d are selected. Currently this is done by an expert, randomly or based on experience resulting in a subset of the data sets $D_{selection}$. With the new approach proposed in this Thesis a subset $D_{potential\ errors}$ will automatically be selected, that contains potential errors. Later in this Thesis, the term subsequence is introduced, that corresponds to one fraction d.

The estimated number of detected faults $N_{detected\ faults}$ is given by

$$N_{detected\ faults} = p(fault|d) * \frac{T_{budget}}{T_{diagnosis}} = p(fault|d) * k \tag{1.1}$$

where $p(fault|d)$ is the probability of a fault in one data set d, T_{budget} is the available time budget, and $T_{diagnosis}$ is the time required for the diagnosis of one data set d. Alternatively k is the number of diagnosed data sets d.

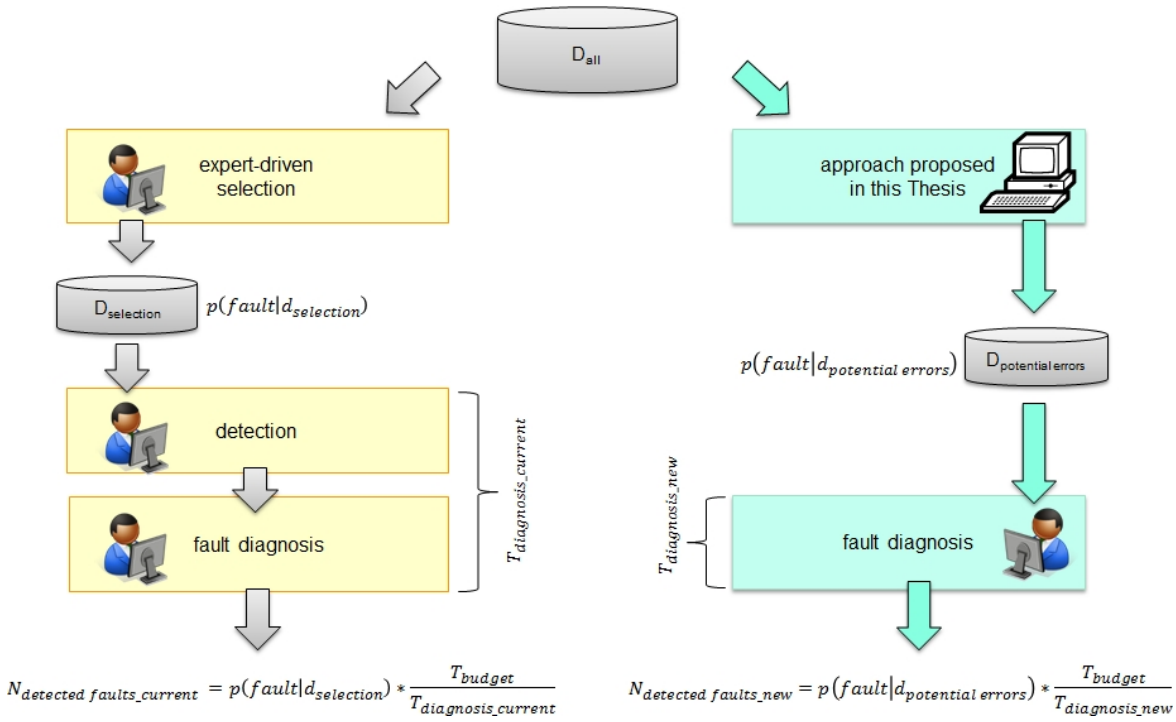


Figure 1.6: Estimated number of detected faults with the current and the new approach ($p(fault|d)$: probability of a fault in one data set d, T_{budget}: available time budget, $T_{diagnosis}$: time required for the diagnosis of one data set d).

Assuming a fixed time budget T_{budget} and considering eq. (1.1), the number of detected faults $N_{detected\ faults}$ can be increased by decreasing $T_{diagnosis}$, corresponding to **Aim 2**. Ways for doing so will be discussed in Chapter 4.

A significant improvement would be if the probability $p(fault|d)$ of finding a fault in a diagnosed data set d could be increased. This will be the focus of this Thesis and will be shown in Chapter 6 to Chapter 9. Instead of randomly selecting $D_{selection}$, the approach in this Thesis automatically selects the subset $D_{potential\ errors}$ which contains potential errors which corresponds to **Aim 1**.

Summarising, the approach in this Thesis will improve $N_{detected\ faults}$ from eq. (1.1) by introducing $T_{diagnosis\_new}$ and $p(fault|d_{potential\ errors})$ such that:

1. $T_{diagnosis\_new} > T_{diagnosis\_current}$

2. $p(fault|d_{potential\ errors}) \gg p(fault|d_{selection})$

The goal of this Thesis is to bridge the gap between testing, which is conducted to a degree that is economically and technically feasible, and the theoretical, but unreachable, optimum of proving correctness for the full system. The underlying idea is not to pre-configure the normal behaviour but rather to extract it from available recordings and then to autonomously detect unexpected deviations.

More specifically, this Thesis addresses the problem of detecting *unexpected occurrences* in recordings from test drives. Unexpected occurrences will be referred to as *anomalies* (see e.g. (Chandola et al., 2009)). Anomalies can be faults that were either not thought of during the design of the testing strategy, arbitrary faults, or faults not modelled due to their complexity or due to constraints of the data acquisition system. Examples are faults in the underlying vehicle, like erroneous sensors or actuators, faults in the software, hardware or parameterisation of electronic control units, or faults in the in-vehicle network. The problem of searching for *known* fault-patterns is not the focus of this Thesis, since this can be done using one of the methods described in Section 1.1.2.

1.5 Related work

Publications specifically addressing the detection of faults in mass data from test drives are rare. Only in rare cases are researchers granted access to the data. However, the detection of faults or anomalies in automotive data has been addressed by a growing number of researchers. The fact that most of the work was published recently shows that the topic gains importance.

In (Suwatthikul et al., 2011) a data-driven approach to classify the health state of an in-vehicle network based on the occurrences of error frames on the CAN bus is proposed. In contrast to this Thesis, (Suwatthikul et al., 2011) bases on a training set of recordings from fault-free and faulty mode.

(Cong et al., 2013) uses anomaly detection on vehicle data in the field of road condition monitoring. Based on a training set of recordings from drives in normal operation mode, potholes are identified as anomalies. Fault detection is not addressed.

In (Mueter and Asaj, 2011) intrusion detection based on recordings from in-vehicle network communication is presented. The underlying assumption is, that the communication on the in-vehicle network has a certain degree of randomness, i.e. entropy. From data recorded in normal operation mode, the normal value of entropy is learnt. An attack, like increasing the frequency of specific messages or message flooding, appears less random and thereby alters the entropy. This way attacks are reported to be detected. As stated in the paper, the value of the normal entropy is likely to depend on the situation the vehicle is in. This is likely to result in a high number of falsely reported anomalies when testing on arbitrary data. This problem is not investigated in the paper, though.

The authors of (Svensson et al., 2008; Byttner et al., 2011) propose fault detection for predictive maintenance of commercial vehicles without modelling effort. As a first step, relevant features in the communication on the in-vehicle network are identified, where close relationship between signals is viewed as relevant. In a second step, data from different vehicles is compared and outliers are detected in an unsupervised manner

being those vehicles deviating from the others. The size of the features is constrained since they are transmitted from vehicles to a central system.

Literature on fault and anomaly detection in general will be surveyed in Chapter 3.

1.6 Conclusion

A vehicle is a high-technology product with a high fraction of vehicle electronics. The gross of innovations is achieved by means of vehicle electronics, which is backed by the fact that the market is growing. Even though massive testing is conducted and research activities are reported on formal verification, there are shortcomings in the analysis of recordings from test drives.

The importance of test drives as a measure for quality assurance is widely accepted. Due to the high costs for conducting of test drives caused by the high fraction of manual work, the recordings of test drives should be viewed as very precious.

Only by effective means of analysing the recordings, one can make sure that the effort put in the conducting of test drives pays off. Existing problems that remain undetected in test drives have a high chance of being detected by customers. The data base of product recalls (Auto Service Praxis, 2013) holds 295 cases for the year 2012 where vehicles had to be recalled and it is uncertain if this list is complete. Consequently, the author believes that effective means of test drive analysis can become a competitive advantage.

1.7 Own Publications

During the writing of this Thesis the following papers were published by the author, with various researchers co-authoring:

- **2010:** In the paper *"Interactive Knowledge Discovery in recordings from vehicle tests"* an approach was proposed to interactively explore recordings from test drives using techniques from the fields of visual data mining and temporal data mining (Theissler et al., 2010).

- **2011:** In the paper *"Interactive Anomaly Detection in time series resulting from local traffic measurements"* it was shown how a visual data mining approach can be used for the offline-analysis of long-term traffic measurements. "Parallel coordinates" were adapted to cope with the spatio-temporal structure of traffic measurements, allowing to interactively explore the data (Theissler et al., 2011).

- **2012:** The paper *"Detecting anomalies in recordings from test drives based on a training set of normal instances"* proposed to use machine learning to support domain-experts by pointing them to the relevant parts in recordings. The idea was validated on data from a test rig with a DC motor (Theissler and Dear, 2012).

- **2013:** The paper *"Autonomously determining the parameters for SVDD with RBF kernel from a one-class training set"* proposed an approach to find the optimal set of parameters for the one-class support vector machine "SVDD" solely based on a training set from one class and without any user parameterisation (Theissler and Dear, 2013b).

- **2013:** In the paper *"An anomaly detection approach to detect unexpected faults in recordings from test drives"* a one-class classification approach is proposed that allows to detect unexpected faults in recordings from test drives without modelling effort (Theissler and Dear, 2013a).

1.8 Outline of the chapters

This Thesis is organised in 11 chapters. Each chapter starts with a brief summary, allowing the reader to rapidly understand the chapter's content. The list of figures and tables can be found at the end of the Thesis. In order to allow for fluent reading, instead of keeping the literature survey and own methodologies in distinct chapters, the author refers to literature where work bases on previous work. For example in Chapter 4 and Chapter 6 literature is surveyed followed by own enhancements.

The following chapters are in this Thesis:

1. **Introduction:** In the current chapter the problem of the complexity of modern vehicles was introduced and shortcomings in the current process of analysing test drive data were identified. Subsequently the aims of the Thesis were presented.

2. **From errors in test drives to faults in the vehicle:** The second chapter discusses why errors are likely to occur in test drives, and surveys potential fault locations in a vehicle.

3. **Anomalies and anomaly detection:** In this chapter the properties of the recordings from test drives are described and the term anomaly is introduced and related to the terms fault, error, and failure. Finally a categorisation of anomalies based on the definition of multivariate time series is given.

4. **Interactive anomaly detection:** A user-driven approach for anomaly detection using visual data mining is introduced in this chapter. It is shown how the analysis of test drive recordings can be improved with adapted visualisation techniques.

5. **Anomaly detection as a classification problem:** This chapter discusses autonomous anomaly detection using classification techniques that learn from training data. The fundamentals of classification theory are given, and common classifiers are introduced. One-class classification is identified to be most suitable for the problem discussed in this Thesis.

6. **One-class classifiers:** In this chapter adaptations of the classifiers k-NN and LOF are compared with the one-class classifier support vector data description (SVDD). SVDD is identified to be most suitable for anomaly detection, though suffers from the problem of having to specify parameters beforehand. In order to solve this problem, a novel parameter tuning approach is proposed.

7. **Enhancing SVDD to multivariate time series:** In this chapter, the classifier SVDD is enhanced to work on recordings from test drives, i.e. on multivariate time series.

8. **The anomaly detection system:** This chapter introduces the anomaly detection system used for test drive analysis. The detection system uses the classification approach proposed in the previous chapter, accompanied by enhanced visual data mining techniques.

9. **Experimental results on recordings from vehicles:** A variety of experimental results on recordings from vehicles are presented in this chapter. The effects of different constitutions of the training set, like integrating data from different drivers and different vehicles, are investigated.

10. **Conclusion:** This chapter summarises the Thesis, identifies the main contributions, and discusses benefits and limitations of the proposed approach.

11. **Outlook:** In the final chapter possible enhancements of the approach are discussed and further research directions are identified.

Chapter 2

From errors in test drives to faults in the vehicle

This chapter emphasises the importance of conducting test drives, discusses the reasons for errors occurring during test drives, and gives an overview of potential fault locations in a vehicle.

This chapter discusses the process of conducting test drives and acquiring data during test drives for later data analysis. Afterwards the reasons are identified why errors are likely to occur during test drives, despite the fact that massive testing effort has been spent in the previous stages. Following that, the terms fault, error and failure are defined. In order to allow for a systematic evaluation of the detection techniques discussed in this Thesis, a categorisation of where in a vehicle faults can be located is presented.

2.1 The process of test drives

This section surveys how test drives are conducted and how the data is recorded using data acquisition systems.

Testing a vehicle under the most realistic conditions by driving it on the road is an integral part of each vehicle manufacturer's process in the development and production of vehicles. In addition to the term test drive, the terms test run, or road trial are commonly used in industry and in literature.

Test drives are conducted by vehicle manufacturers and suppliers throughout various vehicle phases from initial research activities to manufacturing of the vehicle. Hence the vehicle subsystems, for example the ECUs, have different stages in terms of degree of maturity, compatibility and test coverage. The vehicle phases are referred to as research phase, development phase, pre-series phase, and production phase (see Figure 2.1).

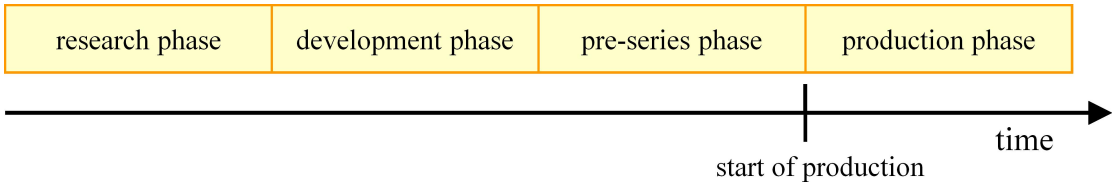


Figure 2.1: Test drives are conducted throughout various vehicle phases ranging from research phase to after start of production

The variety of test drives is enormous, with no two drives being identical. The test drives are conducted by professional test drivers, test engineers, development engineers, researchers, common company employees, in some cases by customers, and sporadically by managers, up to the company's CEO. In addition, at the time of writing this Thesis, driving robots are finding their way into testing vehicles.

The test drives take place either on test tracks or in public traffic, ranging from short-term tests of just a few hundred metres to long-term tests lasting several weeks. In addition, test drives differ in how much the driver is being guided. For example a test drive can be pre-defined by a drive plan subdivided into test cases to be followed

by the driver or the driver does not follow any specifications, aiming to operate the vehicle in everyday use.

During test drives, data is recorded by data acquisition systems for later data analysis. The most common approach is to use data loggers to record the communication on the in-vehicle network. This has the main advantage that, except for some way of accessing the in-vehicle network, the vehicle remains unchanged. At the time of writing, it is common to record the communication from the following network technologies: CAN (Mayer, 2010a), LIN (Mayer, 2010b), FlexRay (Mayer, 2010c), and MOST (Grezemba, 2011).

If the desired data is not available on the in-vehicle network, measurement systems with additional sensors are used to measure e.g. temperatures or voltages. Some prototype vehicles are equipped with additional sensors and a transformer unit sending the measured data over the network, which allows for in-vehicle network data loggers to be used. In addition, some data acquisition systems read internal variables or diagnostic trouble codes (DTCs) from ECUs (Marscholik and Subke, 2008).

2.1.1 Test drives in research phase

During the research phase new vehicle systems, like the concept of a new driver assistance system, are integrated in prototype vehicles (Athanasas and Dear, 2004) or in vehicles of previous model ranges and evaluated in test drives. It lies in the nature of research that faults can be manifold. Some likely faults are incompatible or erroneous software components and unreliable sensors.

The software components are not deployed on the target platform, so timing behaviour that is different to the behaviour expected on the target platform occurs which can lead to errors.

2.1.2 Test drives in development phase

In the development phase the current software versions are continuously tested on the road. Some of the encountered faults are incorrect or incomplete specification or incorrect implementation. Another fault that is often encountered lies in the mapping from a variable in memory to a signal on the bus system. The mapping is done in the ECU software and may operate incorrectly owing to one of the following reasons: an overflow in a signal value due to an insufficient number of assigned bits, an inappropriate scaling factor, or an incorrect mapping algorithm.

2.1.3 Test drives in pre-series phase

Prior to a model ranges' start of production, vehicle fleets are manufactured in the so called pre-series phase and are tested on the road with the aim to assure the quality of the model range as well as of the production and testing processes (Friedrich et al., 2009).

The vehicles are partly hand-crafted, the ECU software and its parameterisation are manually downloaded to the ECUs. Hence, faults in the versions or variants of an ECU's software, or parameterisation of an ECU that is incompatible to further ECUs in the network are likely faults.

2.1.4 Test drives in production phase

Accompanying the production phase, test drives are conducted with series vehicles after start of production. In order to assure quality of the vehicles, a fraction is selected for test drives. In some cases, field data is collected from customers of leased electric and fuel-cell vehicles and from fleets of commercial vehicles.

Likely faults to occur in production phase are manufacturing faults like improper wiring, e.g. open or short circuits, or incompatible software versions or variants.

2.2 Why are errors likely to occur during test drives?

This section discusses why errors are likely to occur during test drives, even though intensive testing has been done on the vehicle's individual components prior to a test drive.

Despite massive testing effort prior to integration of the components in the vehicle, errors are yet likely to occur during test drives. The following key reasons were identified:

1. final integration of all components takes place in the vehicle
2. incomplete simulation models for test drives
3. the automotive development process
4. the supply chain in the automotive industry

2.2.1 Final integration of all components in the vehicle

All individual vehicle components have been tested in laboratory environment, yet the integration of all ECUs, sensors and actuators in the vehicle puts the components under real conditions for the first time. The previous tests are based on specifications and implicit assumptions using test cases, simulations, and test rigs.

It is in the vehicle, where problems from the likes of incompatible software versions or parameter sets, or with the integration of new technology, occur. In the vehicle there are no assumptions about subsystems and no simulations.

2.2.2 Incomplete simulation models for test drives

The vehicle's components are tested with simulation models. However the variety of test drives is not covered by the models.

Test drives are conducted in a wide range of different conditions ranging from very hot to very cold climate, with bad road conditions, differing traffic flow, heavy load, and different driver behaviour.

One type of tests is to operate the vehicle under defined harsh conditions to cover the expected vehicle lifetime within a much shorter period of time (Friedrich et al., 2009), which means that e.g. wear-out of mechanical components can be encountered even for pre-series vehicles that are only a few months old.

2.2.3 The automotive development process

The development cycles of vehicle models are shrinking and it is economically not reasonable to follow a strictly sequential development process. Hence, components under development are tested against other software or hardware components which themselves are in a pre-series stage.

One approach of testing prototype vehicle subsystems on the road at an early stage is to integrate the new components into vehicles of previous models that are similar but not fully identical to the target vehicle.

In the development of new models or for model upgrading, manufactures do not develop the model ranges from scratch. They rather use and adapt parts of existing models, i.e. some components will be used from previous vehicles. One approach is to integrate the new components in the environment of previous vehicles, which are likely to be not fully compatible to the newly developed vehicle subsystems.

Components are developed in accordance to a specification, but in general, the software and hardware versions and the specification of components integrated in the series vehicle evolves over the development and pre-series phase.

Generally it can be observed that the implementation of an in-vehicle network is never an optimal solution from the viewpoint of software or electronics. It is rather a compromise of many additional interdisciplinary factors such as cost, weight of wiring, space for wiring and ECUs, legislative obligations and the structure of the supply chain.

2.2.4 The supply chain in the automotive industry

Contemplating the structure of the automotive industry, it becomes obvious that the gross of vehicle functions are not developed by vehicle manufacturers, but rather by suppliers according to the manufacturers' specification by means of subcontracts. A typical supply chain has multiple tiers of suppliers. The first tier supplier supplies a vehicle function like e.g. the adaptive cruise control system, but in turn procures software and hardware components from further, more specialised suppliers and integrates them into the vehicle subsystem.

There is a number of different scenarios of supply chains ranging from one supplier supplying all components to a scenario where sensors, actuators, ECU hardware, ECU base software, and ECU application software are supplied by different suppliers. In addition, for vehicle functions using data from multiple sensors, the different sensors are likely to be supplied by different suppliers.

In the integration tests, manufacturers treat ECUs as black boxes and only view them through the specified interface or access data via standardised diagnostic services (Marscholik and Subke, 2008).

2.3 Faults, errors and failures according to ISO 26262

The behaviour of a system not acting as expected is colloquially described as a fault, a failure, a malfunction, or an error. The definition of exact terms is controversially discussed in the literature. In this Thesis, the terms *fault*, *error*, and *failure* are used as defined by ISO 26262-1 (ISO 26262-1, 2011). A fault may manifest itself as an error, which in turn may cause a failure (Sommerville, 2001). Faults can be subdivided into permanent, intermittent and transient faults.

Definition 2.1 Fault*: A* fault *is an "abnormal condition that can cause an element or an item to fail" (ISO 26262-1, 2011).*

Definition 2.2 Error*: An* error *is the "discrepancy between a computed, observed or measured value or condition, and the true, specified or theoretically correct value or condition" (ISO 26262-1, 2011).*

Definition 2.3 Failure*: A* failure *is the "termination of the ability of an element, to perform a function as required" (ISO 26262-1, 2011).*

An example is an incomplete branch statement in the source code of an ECU's software, which corresponds to a *fault*, in this case to a software bug. This fault is permanently present in the vehicle, but will not necessarily ever reveal itself. Only specific input parameters, for example values read from a sensor, that are not properly covered by the branch statement lead to an observable deviation from expected behaviour. This deviation corresponds to an *error*, which in turn may cause a *failure*, e.g. a driver assistance system stops to function.

2.4 Fault locations

After having discussed the reasons for errors occurring during test drives and having defined faults, errors and failures, this section identifies a categorisation of where in

a vehicle faults can occur. From an observed error, the next step is to identify and locate the fault that caused the error. This process is commonly referred to as fault detection and diagnosis (FDD) (Isermann, 2006).

A coarse categorisation of fault locations can be found in (Suwatthikul, 2008), where the three levels network, feature, and component are used. The component level contemplates each network node individually, the feature level contemplates individual vehicle functions and the network level considers the entire inter-networking. In (Jeutter, 2008), the author categorises faults with respect to the layers of the Open Systems Interconnection model (OSI). The physical level is concerned with the electrical signals on the wire, the protocol level analyses the data flow, and the data level contemplates the data as seen by the ECUs and finally the system level is concerned with vehicle sub-systems. In addition, the timing is considered across all levels ranging from physical to system level. The authors in (Stein et al., 2006) subdivide fault locations in vehicle electronics into sensors/actuators, electric, buses, and ECUs. In this section an own, more detailed categorisation of fault location is given, that is partly based on (Stein et al., 2006).

2.4.1 Example of a data flow

An example of a data flow from reading sensor data to controlling an actuator is shown in Figure 2.2. The components in this example are either directly powered by a power supply or as in the case of the sensor and the actuator by an ECU. As illustrated in Figure 2.2, the components are interconnected by cables and connectors.

Signals of “vehicle subsystem 1” are measured by a sensor. The sensor data is read and processed by “ECU1”, embedded into a data frame and transmitted over the sub-network “bus 1” via the gateway to the sub-network “bus 2”, where the data frame is read by “ECU2” and the signal values are restored. Based on the input value from “vehicle subsystem 1”, “ECU2” controls an actuator which in turn influences “vehicle subsystem 2”. In order to allow for analysis, the data transmitted over the two bus systems is recorded by a data acquisition system.

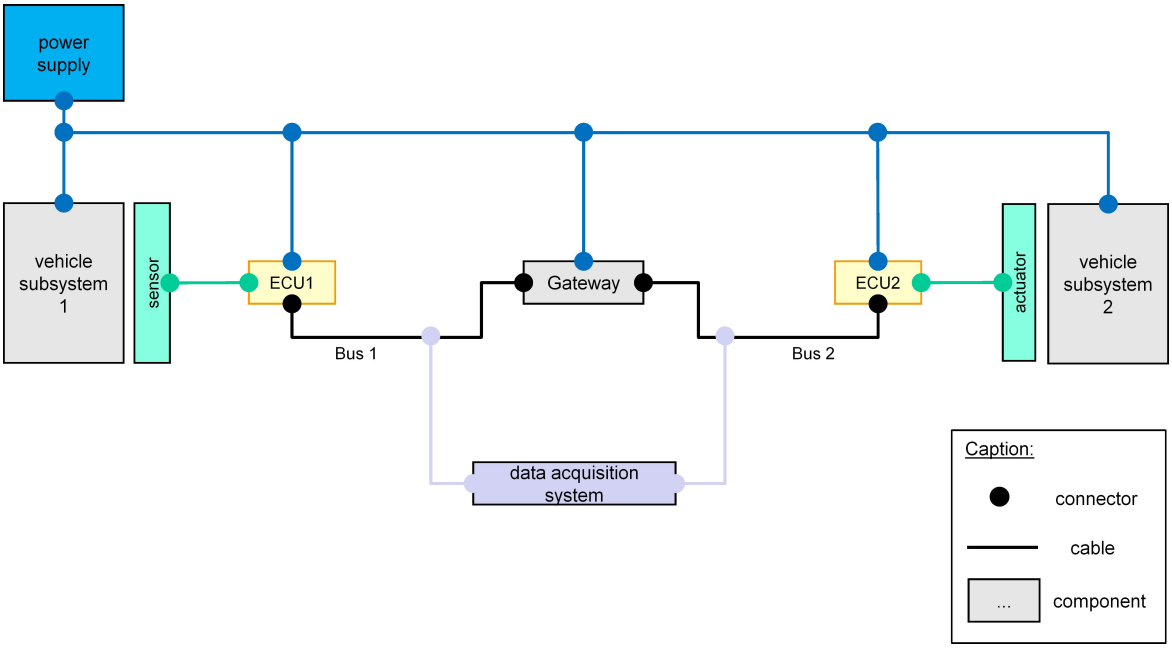


Figure 2.2: Fault locations in an in-vehicle network

2.4.2 Categorisation of fault locations

In the described data flow, some potential fault locations are evident, like faults in the connectors or cables, or faults in an ECU's software. The identified locations are described in the rest of this section and a categorisation is given in Figure 2.3. At the first level the locations are categorised into function specifications, network, sensors, actuators, ECUs, gateways, power supply, vehicle subsystems, and the data acquisition system.

The author refers to a function specification as the specification of a vehicle function, which is any built-in functionality offered by the vehicle. A vehicle function can range from something as simple as the indicator or the windscreen wipers to complex systems like adaptive cruise control.

The network as a potential fault location includes the cable harness, the physical topology and the network's configuration. The cable harness is composed of connectors and cables which in turn consist of wires and shielding. Faults can be from the likes of open

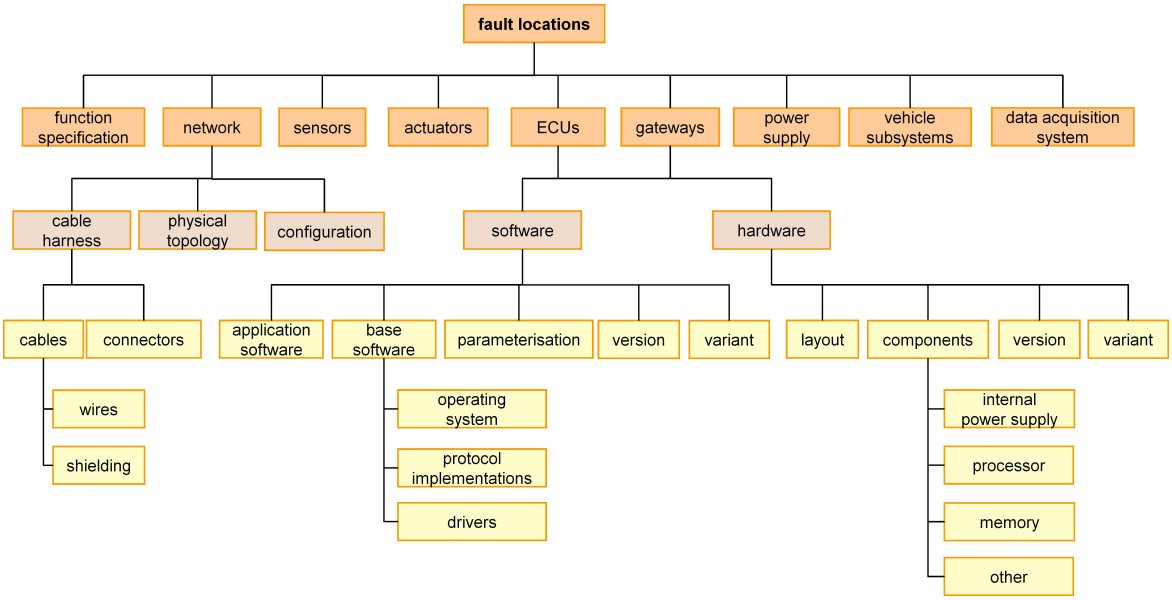


Figure 2.3: Categorisation of fault locations in an in-vehicle network

circuits, short circuits to either battery voltage or ground, or short circuits between wires (Krauß, 2010). Faults located in the cables can be caused by electromagnetic interference due to improper or damaged shielding. Concerning the connectors, incorrect pin assignments is a common problem in early vehicle phases. Improper scheduling of messages or implementation faults can lead to any of the following errors: lost messages, too many messages (Suwatthikul, 2008), delayed or corrupted messages.

The physical topology is the layout of the network and describes how many nodes (ECUs, gateways, sensors and actuators) are in the network, the partitioning of the network into sub-networks using gateways, and the used network technologies. The author refers to the network configuration as any settings of the network that do not affect the physical topology. Depending on the type of bus system the network's configuration consists of different settings, examples are cycle times and priorities of messages in the case of CAN, or number of slots per cycle in the case of FlexRay.

Faults occur in sensors and actuators, which are either directly wired to the ECUs as analogue or digital input/output signals or by means of a bus system. A defective sensor may reveal a change of gain or an offset (Isermann, 2006). Also either no values, or values that are out of the valid range can be supplied by a sensor. More difficult to

detect is a scenario where a sensor supplies erroneous values, that are within the valid range, but are implausible with respect to further input values (Krauß, 2010). Faults located in the actuators can be an invalid calibration or wear-out.

In terms of ECUs one distinguishes between physical and virtual ECUs. Due to space and weight restrictions, several virtual ECUs are sometimes placed in one enclosure. The virtual ECUs are hence connected to the same power supply and to the same bus system. So in addition to the visible physical ECUs in the physical layout of an in-vehicle network as shown in Figure 1.1, additional virtual ECUs can be found in a vehicle.

Further potential fault locations are an ECU's software or hardware. An ECU's software can be further categorised into (1) application software, (2) base software, i.e. the operating system and implementation of protocols and drivers, and (3) parameterisation, which is composed of characteristic curves and coding parameters adapting the vehicle to extra equipment or to a region, e.g. right- or left-hand-drive. The parameterisation can either be incorrect or incompatible to the vehicle. Further fault locations are the versions and variants of the ECU software. In isolated tests the software versions may function properly, but the software versions of ECUs are incompatible among each other. Similarly, faults occur in the variants of the software, if for example the variant is not compatible to the vehicle's extra equipment or to the deployed hardware.

In addition to faults in the ECU software, the ECU's hardware is a potential fault location. The hardware can be subdivided into the ECU's layout and the ECU's internal components, where giving an exhaustive list is not relevant for this Thesis. The most relevant fault locations are internal power supply, the processor and the memory. In addition, as with software components, faults in the ECU's hardware can be due to incorrect versions and variants. Additionally there can be incompatibilities of software and hardware versions and variants.

Gateways connect the different sub-networks and are ECUs themselves. In practice, the gateway functionality is usually embedded as a virtual ECU into a physical ECU. Yet, the distinction between gateway and ECU is made in this Thesis because faults

in the gateway functionality can lead to entire sub-networks not being able to communicate. Faults in a gateway would be the permanent or sporadic violation of timing constraints in the forwarding of messages, lost messages, or improper data transformation between sub-networks using different technologies.

As shown in Figure 2.2, the vehicle's components are powered by the vehicle's power supply, which typically is the battery voltage. Hence this is a central location for faults. Finally, mechanical vehicle subsystems interfacing with vehicle electronics are potential fault locations. Faults in vehicle subsystems, which for example are the engine or the braking system, can be observed in the signals measured by sensors.

The data is recorded by data acquisition systems, which can potentially induce faults into the vehicle or into the recording. An example of the latter one would be messages exchanged by ECUs via the in-vehicle network that were sporadically not properly recorded due to a malfunction of the data acquisition system.

2.5 Conclusion

In this chapter it was shown that test drives are conducted throughout the entire vehicle life cycle. Despite massive testing effort prior to the integration of the components, test drives are inevitable. Errors occur during test drives since a test drive is a test under the most realistic conditions.

From an observed error, the underlying fault can be identified. As discussed, the potential fault locations in a vehicle are manifold. Faults can be present in specification, software, hardware, and wiring. General techniques to detect faults or anomalies in data will be surveyed in the next chapter.

Chapter 3

Anomalies and anomaly detection

This chapter describes the properties of recordings from test drives, introduces the term anomaly and relates it to the terms fault, error, and failure. It further categorises anomalies in multivariate time series into three types and surveys anomaly detection techniques.

This Thesis is concerned with the detection of anomalies in recordings from test drives. The recordings are viewed as multivariate time series data. The terms time series and anomaly as used in this Thesis will be defined in this chapter. Following that, anomalies in multivariate time series are categorised into three types.

3.1 Recordings from test drives: Time series data

This section describes the properties of the recordings from test drives. The recordings are obtained by data acquisition systems and contain either signals or raw messages from which signals can be extracted. The messages and signals are time stamped,

hence a recording is viewed as time series data, as it can easily be transformed to time series.

One recording contains a variety of signals where the number can be in the region of several thousand. Some signals have interdependencies, for example the vehicle speed signal typically depends on the signal holding the accelerator pedal's current position.

Definition 3.1 Recording*: A* recording *refers to discrete time-stamped signal values recorded during tests. A recording may contain multiple signals.*

Figure 3.1 shows an excerpt of a test drive recorded by a data acquisition system [1]. The signals speed, steer angle and engine rpm were extracted from the recorded in-vehicle network communication.

A recording corresponds to time series data or can be resampled equidistantly to become time series data. Time series can be *univariate* or *multivariate* (Mitsa, 2010), where observations of one variable are referred to as univariate time series and observations of multiple variables are referred to as multivariate time series. An example of a univariate time series is the recording of the vehicle's speed signal over some period of time. A multivariate time series would be the vehicle's speed together with further signals like yaw rate, steering wheel angle and engine speed.

Definition 3.2 Univariate time series*: A* univariate time series X_T *is a finite sequence of* N *data points ordered by time. For every time point* t_i *in* $\vec{T}$ *there exists one data point* x_i *in* $\vec{X}$*, where* $1 \leq i \leq N$*. The distance* $\Delta t = t_i - t_{i-1}$ *is finite and equidistant for all* $i > 1$*. A single data point is denoted by* x_{t_i}*.*

$$X_T : \vec{T} \mapsto \vec{X} \quad with \quad \vec{T} = (t_1, t_2, \ldots, t_N) \quad and \quad \vec{X} = (x_1, x_2, \ldots, x_N) \tag{3.1}$$

[1] Recorded and visualised using "Tedradis" by IT-Designers GmbH (Koch and Theissler, 2007; Marscholik and Subke, 2008)

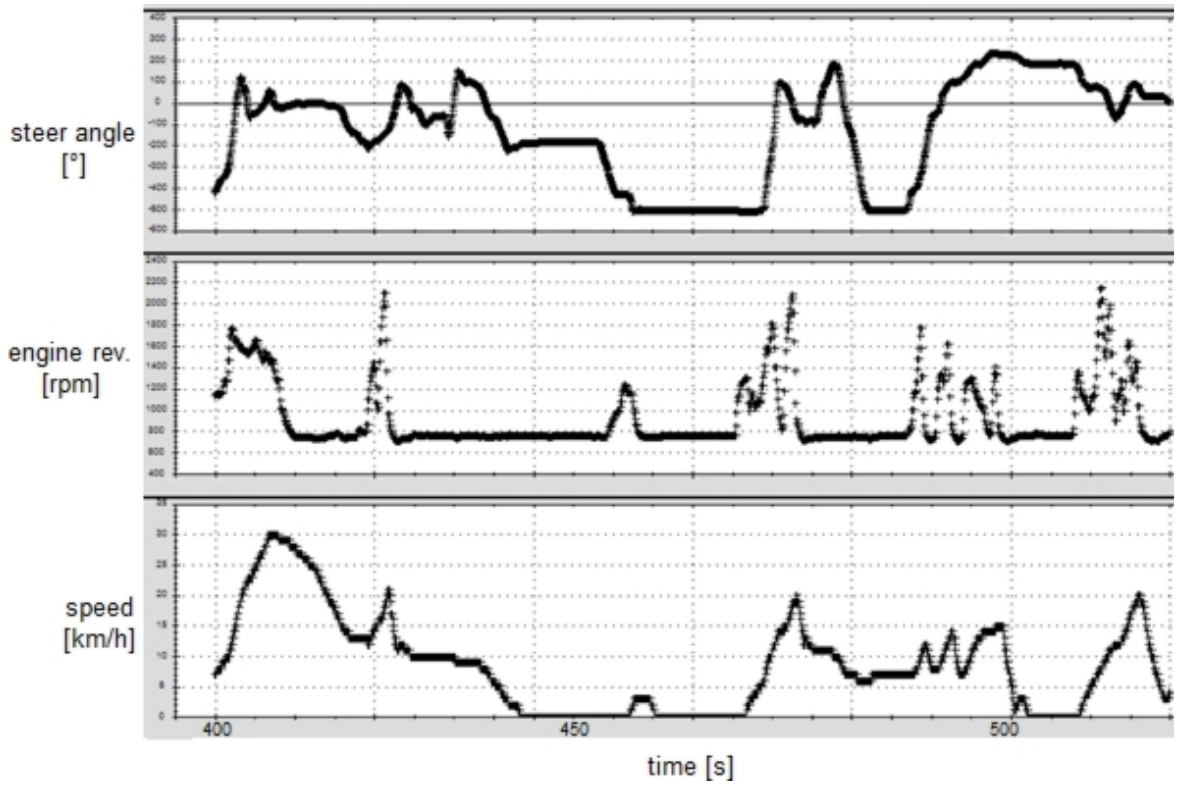


Figure 3.1: Time series data extracted from two minutes of in-vehicle network communication recorded during a test drive showing the steer angle, engine rpm, and vehicle speed.

Definition 3.3 Multivariate time series*: A* multivariate time series Y_T *consists of at least two univariate time series, which are denoted by* X_{T_u} *with* $1 \leq u \leq M$*. The number of data points* N *and the vector of time stamps* $\vec{T}$ *is identical for all* X_{T_u}*. Hence for every time point* t_i *there exist* M *data points. A single data point is denoted by* x_{u,t_i}*.*

$$Y_T = (X_{T_1}, X_{T_2}, \ldots, X_{T_M}) \tag{3.2}$$

which can be rewritten in matrix form

$$Y_T = \begin{pmatrix} x_{1,t_1} & \dots & x_{1,t_N} \\ \vdots & \ddots & \vdots \\ x_{M,t_1} & \dots & x_{M,t_N} \end{pmatrix} \tag{3.3}$$

Univariate time series contained in a multivariate time series are possibly, but not necessarily, correlated. So X_{T_1} might be input data from a sensor, X_{T_2} a calculated time series based on the sensor data and X_{T_3} the resulting output data transmitted to a connected actuator.

Working with time series data, the entire time series, parts of the time series or individual data points can be contemplated. Therefore the terms *subsequence* and *sets of subsequences* are introduced.

Definition 3.4 Subsequence*: A* subsequence *is any univariate time series having* 1 *to* N *consecutive data points contained in a univariate or multivariate time series. In a univariate time series a subsequence shall be denoted by* $X_{t_j \dots t_k}$ *with* $1 \leq j \leq k \leq N$ *and in a multivariate time series it shall be denoted by* $Y_{u,t_j \dots t_k}$ *with* u *indexing the univariate time series.*

$$X_{t_j \dots t_k} : (t_j, \dots, t_k) \mapsto (x_j, \dots, x_k) \quad with \quad 1 \leq j \leq k \leq N \tag{3.4}$$

Following definition 3.4, a subsequence of a time series X_T can either be an individual data point, some consecutive data points, or the entire univariate time series X_T.

Each subsequence is fully addressed by its first point, indexed by j, length W, and in the case of multivariate time series by the index of the univariate time series u. This way a subsequence can be allocated to its exact position in the time series it originated from.

In addition it is appropriate to be able to describe a group of subsequences:

Definition 3.5 Set of subsequences*: A* set of subsequences *is denoted by S. The contained subsequences are denoted by* s_i *with* $i = 1, ..., |S|$*, i.e.* $S = \{s_1, s_2, ..., s_{|S|}\}$.

3.2 Defining anomalies

Searching a data base of recordings from test drives for previously defined fault-patterns, e.g. values that are out of the valid value range, is a challenge due to the vast amount of data. However, the problem of searching for known fault-patterns is not the focus of this Thesis, since this can be achieved by techniques that have been extensively researched, e.g. pattern matching.

This Thesis addresses the problem of detecting *unexpected occurrences* in a data base holding recordings from test drives. Unexpected occurrences are referred to as *anomalies* (Chandola et al., 2009). An anomaly may be observed due to an error, and may point an expert to a fault in the vehicle. While an error is always viewed as an anomaly, the opposite is not true (Figure 3.2).

The term anomaly is used for occurrences that do not conform to normality, where normality is in no way absolute. What is "normal" can be learnt from a training data set, be described using models or be expressed by a domain expert. In ISO-26262 (ISO 26262-1, 2011) the term anomaly is defined as a "condition that deviates from expectations, based, for example, on requirements, specifications, design documents, user documents, standards, or on experience". Other terms corresponding to the term anomaly used in literature are novelty, outlier (Hodge and Austin, 2004) and discord (Keogh et al., 2006). The description of anomalies from (Chandola et al., 2009) is adapted in order to describe anomalies in multivariate time series.

Definition 3.6 Anomaly*: In this Thesis an* anomaly *is a deviation in the behaviour of a univariate time series or in the relationship of multiple univariate time series from expected behaviour.*

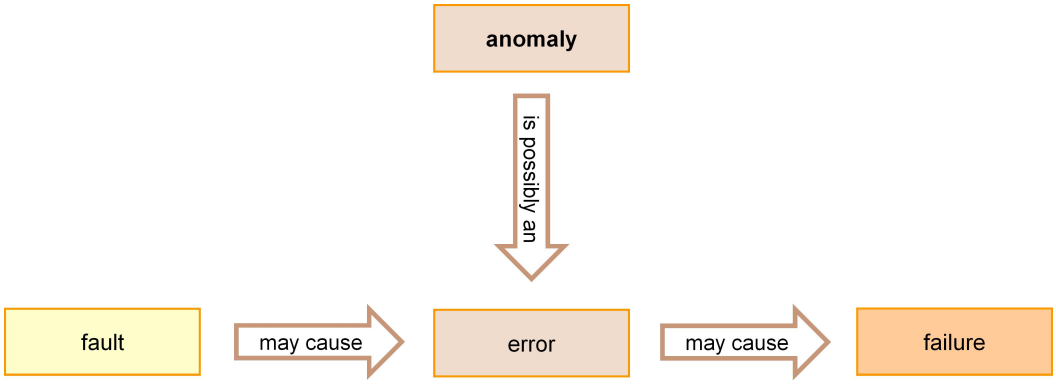


Figure 3.2: An anomaly is a potential error. An error is caused by a fault and may cause a failure.

Anomalies could be detected manually, semi-autonomously, or fully autonomously. Generally speaking, subsequences are classified by a function f_a as normal or abnormal, as defined in definition 3.7.

Definition 3.7 *Let f_a be a function that, applied on a set S, returns 1 if S is normal or empty and -1 if S is abnormal.*

$$f_a(S) = \begin{cases} +1 & : \textit{if } S \textit{ is normal or } S = \emptyset \\ -1 & : \textit{if } S \textit{ is abnormal} \end{cases} \tag{3.5}$$

3.3 Categorising anomalies in multivariate time series

The types of anomalies in multivariate time series are categorised in this section. A distinction is made whether the anomaly occurs either in one of the individual univariate time series or in their relationship.

3.3.1 Anomaly types

In (Chandola et al., 2009), anomalies are categorised as point, contextual, and collective anomalies. The idea of point and contextual anomalies was borrowed and applied to multivariate time series.

As defined in definition 3.3, a multivariate time series consists of multiple univariate time series. Therefore in a multivariate time series, anomalies of a univariate time series can occur. Additionally, in multivariate time series, anomalies in the relationship between the contained univariate time series can occur.

The following three types of anomalies, also shown in Figure 3.3, are distinguished in this Thesis:

1. *Type 1: subsequence anomaly in univariate time series:* An individual subsequence $X_{t_j \dots t_k}$ that can be classified as normal or abnormal without contemplating further subsequences. This can, for example, be done by comparing the subsequence with some threshold defining the border between normal and abnormal. Any subsequence violating the valid value range is a subsequence anomaly.

2. *Type 2: contextual anomaly in univariate time series:* An individual subsequence $X_{t_j \dots t_k}$ that cannot be classified as normal or abnormal. The classification requires knowledge about the context, i.e. about other subsequences within the same univariate time series, for example neighbouring subsequences like $X_{t_{j-2} \dots t_{j-1}}$ and $X_{t_{k+1} \dots t_{k+2}}$.

3. *Type 3: contextual anomaly in multivariate time series:* An individual subsequence $Y_{1,t_j \dots t_k}$ in a multivariate time series Y_t that cannot be classified as normal or abnormal. The classification requires knowledge about the context, i.e. about subsequences of additional univariate time series within Y_t, like $Y_{2,t_j \dots t_k}$. The set S containing the subsequences $Y_{1,t_j \dots t_k}$ and $Y_{2,t_j \dots t_k}$ is an anomaly.

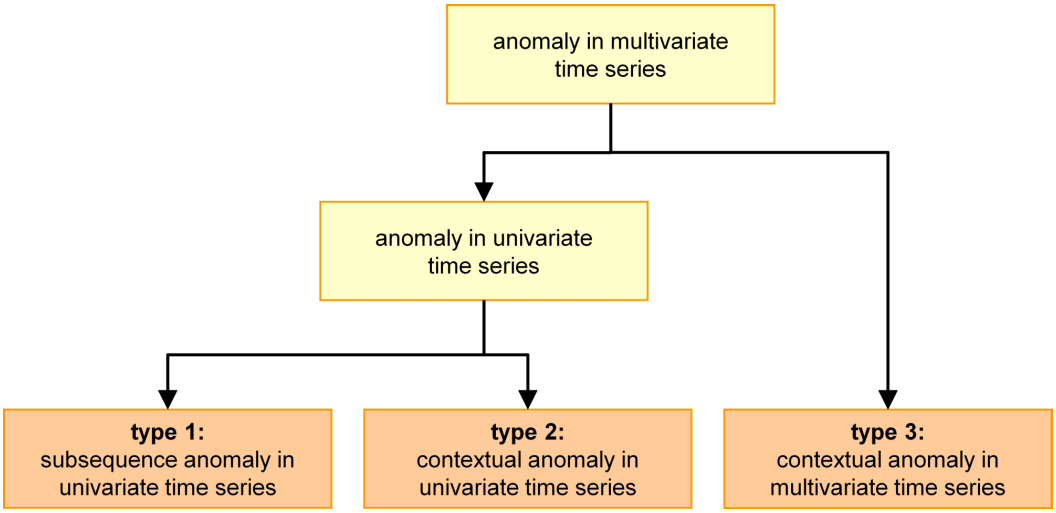


Figure 3.3: Categorisation of anomalies in multivariate time series.

A subsequence anomaly is an individual subsequence, while a contextual anomaly is a set of subsequences.

Furthermore, if there exists knowledge about causal dependencies between the univariate time series, a set S may be partitioned into the contextual set S_c and the indicator set S_i, i.e. $S \rightsquigarrow pair(S_i, S_c)$ with $S_i \cup S_c = S$.

Definition 3.8 Contextual set*: A* contextual set *is a set holding contextual information and is denoted by* S_c*. The definition of context is done a priori.*

Definition 3.9 Indicator set*: An* indicator set *is a set that requires consideration of a contextual set and is denoted by* S_i*.*

When $S \rightsquigarrow pair(S_i, S_c)$ and $f_a(pair(S_i, S_c)) = -1$, following statements about S can be made:

1. If $S_i \neq \emptyset$ and $S_c \neq \emptyset$, then S is a contextual anomaly (type 2 or 3).

2. If the number of elements $|S_i| = 1$ and $S_c = \emptyset$, then S is a subsequence anomaly (type 1).

3.3.2 Examples

This section gives examples for each of the three types of anomalies given in the categorisation in Figure 3.3. In order to demonstrate the proposed categorisation of anomalies with a real-world data set, the test rig shown in Figure 3.4 was built. A brushless DC motor[2] was mounted on a retainer with a miniature wheel attached. A brushless DC motor was chosen, since this type of motor is widely used in vehicles e.g. as auxiliary units like automatic seat adjustment or the window opener. More powerful electric motors of this type are used as part of the power train in hybrid, pure electric or fuel-cell vehicles. This test rig is also used for experiments in later chapters.

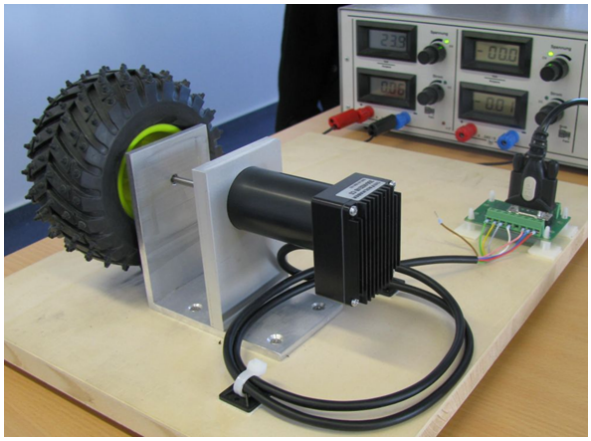


Figure 3.4: DC motor test rig.

The motor was operated in servo mode, meaning it runs in closed-loop control mode. Observing the signals, no external sensors are needed. It is based on the motor's input and output signals only, referred to as sensorless detection, see e.g. (Moseler and Isermann, 2000). The following procedure was executed by the test rig, while the motor's position and the DC current were recorded.

1. 10 revolutions in forward direction

2. hold position for 10 seconds

[2]Faulhaber GmbH & Co. KG: Series 3564. Brushless DC servo-motor with integrated motion controller and RS232 interface.

3. 10 revolutions in backward direction

4. hold position for 5 seconds

5. goto step 1

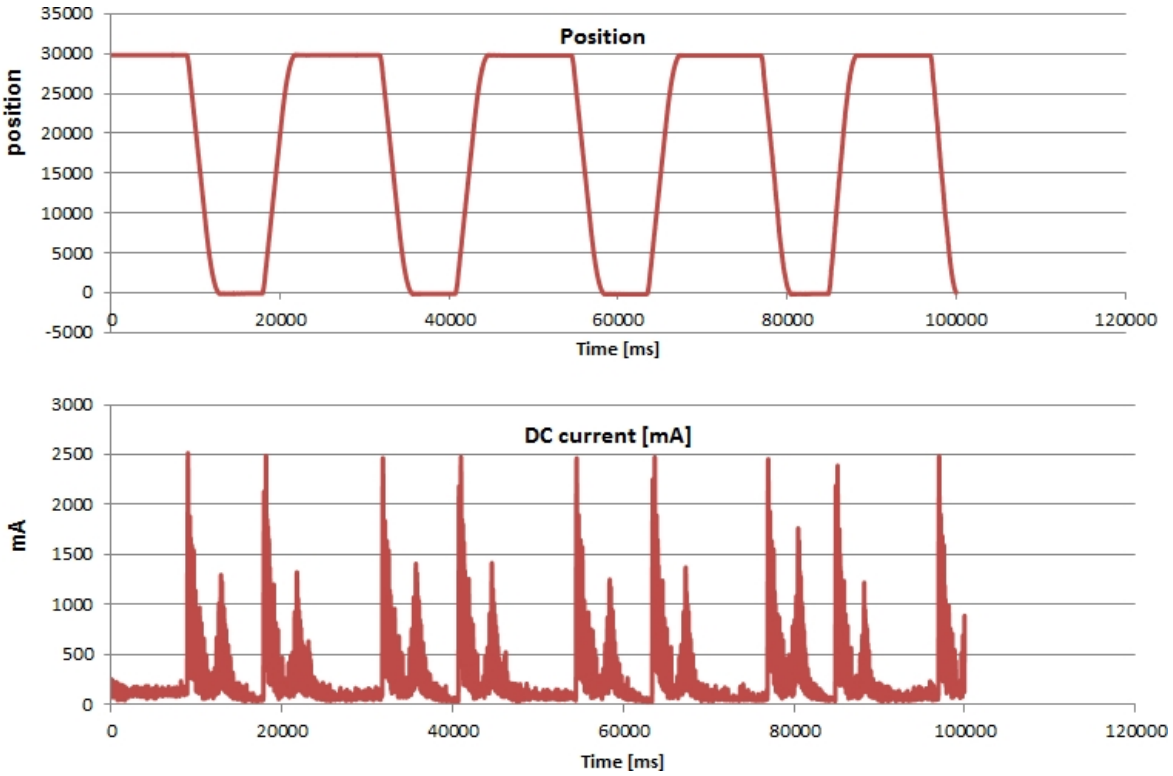


Figure 3.5: Position and DC current of a DC motor in normal operation mode.

The data recorded in normal operation mode is viewed as a reference data set. The position and DC current signal of the motor in normal operation mode over a period of approximately two minutes are shown in Figure 3.5. A position value of 3000 corresponds to one revolution, i.e. 360 degrees. It can be seen that peaks in the DC current are correlated with the acceleration and deceleration of the motor, where the high peaks correspond to the motor accelerating and medium peaks to the motor decelerating.

The following examples of anomalies are subsequences or sets of subsequences deviating from the behaviour shown in the reference data set. The anomalies were manually injected by varying the load on the motor.

3.3.2.1 Subsequence anomalies in univariate time series

The following normal behaviour for the DC current signal can be deduced from the recordings in normal operation mode shown in Figure 3.5:

1. the minimum normal DC current value is 27 mA

2. the maximum normal DC current value is 2514 mA

Figure 3.6 shows a recording of the DC current over a period of 100 seconds. The two dashed lines represent the valid value range. The subsequence s_1 marked in Figure 3.6 is obviously not within this range, and is therefore an anomaly. Since no further subsequences need to be contemplated in order to make the decision, the marked subsequence s_1 is an anomaly of type 1 (subsequence anomaly in univariate time series). The anomaly was injected by blocking the motor for a short period of time.

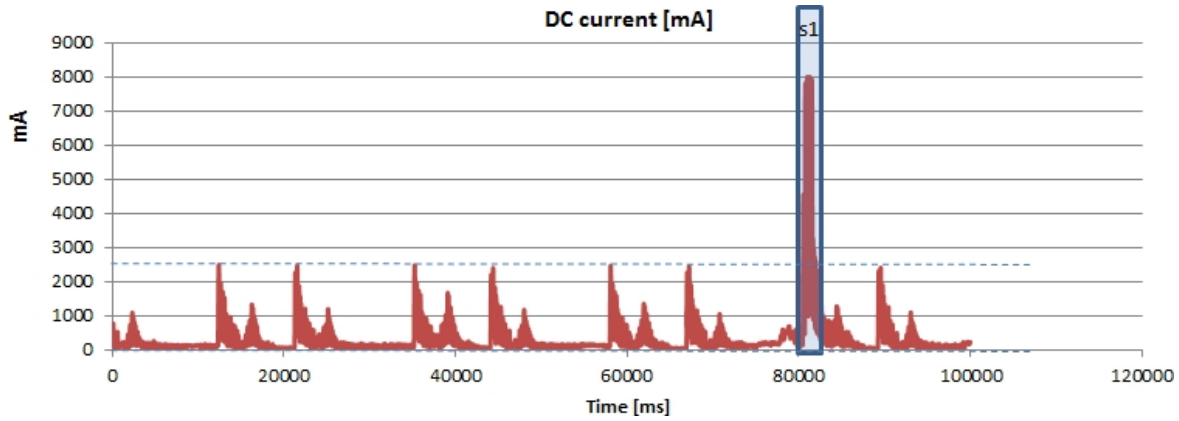


Figure 3.6: Example of a subsequence anomaly in univariate time series (type 1). The values of subsequence s_1 exceed the valid value range.

3.3.2.2 Contextual anomalies in univariate time series

Another recording of the DC current signal is shown in Figure 3.7. The signal could be segmented into subsequences of the following types: *low values*, *medium peak* and *high peak*. The following rules can then be deduced from the reference data set:

1. a *high peak* is always preceded by a subsequence with *low values*

2. a *medium peak* is always preceded by a *high peak*

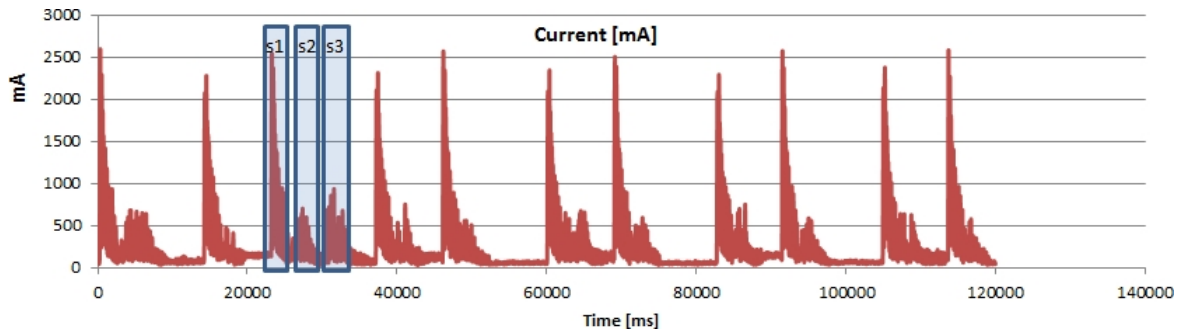


Figure 3.7: Example of a contextual anomaly in a univariate time series (type 2) in the DC current of the motor. The set $S = \{s_2, s_3\}$ is abnormal.

In Figure 3.7 the subsequence s_2 (*medium peak*) is preceded by s_1 (*high peak*). In accordance with the second learnt rule, s_2 is therefore normal. The subsequence s_3 (*medium peak*) is preceded by s_2 (*medium peak*) violating the first rule. The set of subsequences $S = \{s_2, s_3\}$ is therefore a type 2 anomaly (contextual anomaly in univariate time series).

The anomaly was injected by manually trying to readjust the wheel attached to the motor, which leads to an increase in the DC current signal in order for the motor to hold its position.

3.3.2.3 Contextual anomalies in multivariate time series

In addition to the two types of anomalies that may occur in individual univariate time series, the relation between the signals of the multivariate time series recorded from the test rig can be abnormal.

In the reference data set in Figure 3.5 it can be seen that subsequences with constant position values occur coinstantaneously with subsequences with low values of the DC current. The following correlations can be observed:

1. an acceleration in the position signal leads to a *high peak* in the DC current
2. a deceleration in the position signal leads to a *medium peak* in the DC current
3. a subsequence with *constant values* of the position signal leads to a subsequence of *low values* in the DC current

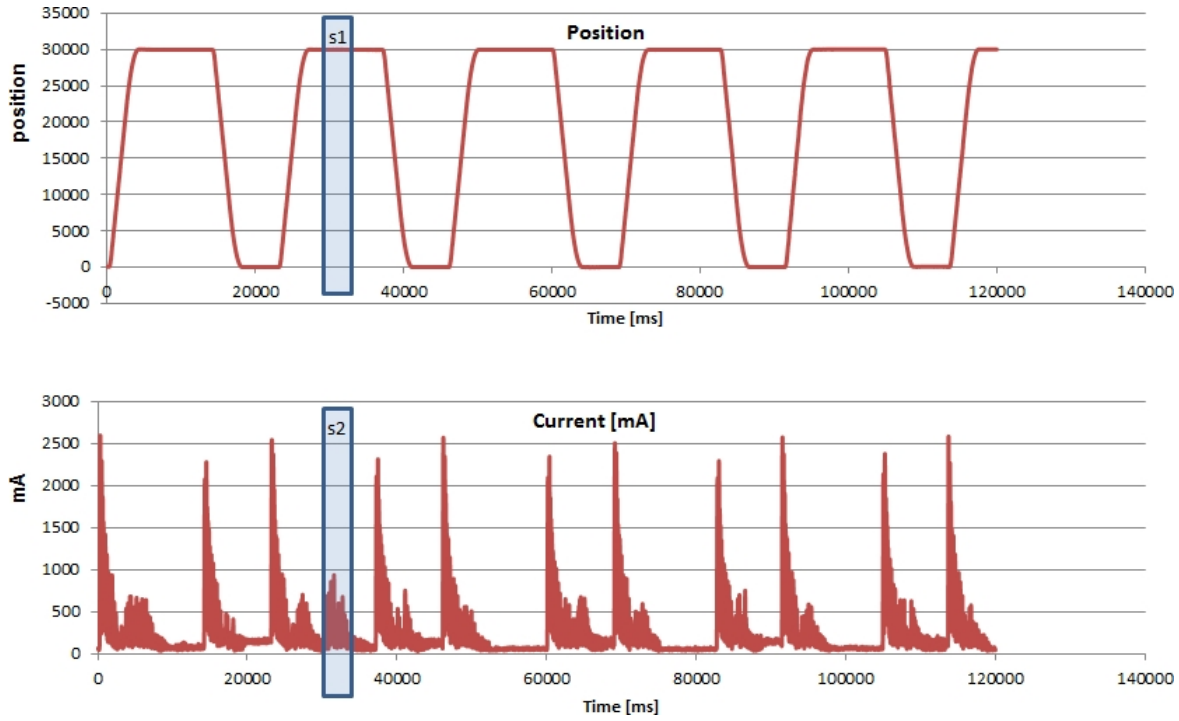


Figure 3.8: Contextual anomaly in the dependency within the multivariate time series.

While s_1 in Figure 3.8 is a subsequence of *constant values*, s_2 shows a medium peak, violating the third learnt rule. Therefore the set $S = \{s_1, s_2\}$ is an anomaly of type 3 (contextual anomaly in multivariate time series).

3.3.3 Discussion

The categorisation of anomaly types is considered essential in order to evaluate anomaly detection techniques. Depending on the type of anomaly, different anomaly detection techniques are required. One approach is to focus on one type of anomaly. If no knowledge exists about the types of anomalies expected in a data set, a combination of different detection techniques needs to be used in order to be able to detect all types.

The idea to integrate some kind of additional information like a context was addressed by several researchers. In (Song et al., 2007) the term conditional anomaly is used rather than contextual anomaly. It is argued that integrating contextual knowledge improves the chance that reported anomalies are interesting for the user. It is proposed, to have the user partition the input data into environmental and indicator attributes prior to anomaly detection, where environmental attributes correspond to the contextual attributes in (Chandola et al., 2009) and indicator attributes correspond to the behavioural attributes in (Chandola et al., 2009). In (Hauskrecht et al., 2010) an approach for clinical alerting is proposed that, in addition to variables like medication given at certain points in time, considers the condition of the patient at that time.

Subsequence anomalies are the easiest to detect, since detection is based upon comparison of one instance against some threshold. One approach in anomaly detection is trying to transform complex anomalies to subsequence anomalies, which allows for simpler anomaly detection mechanisms. Contextual anomalies for example can be transformed into subsequence anomalies, if the dependencies between the univariate time series can be mapped to features.

3.4 Anomaly detection

As anomalies may point to potential faults, literature about fault detection is surveyed in this section to illuminate the techniques used in this research field followed by techniques that could be used for fault detection.

According to (Venkatasubramanian et al., 2003c) fault detection is always based on some kind of redundancy. Hardware redundancy uses redundant signals in a system to check for plausibility, e.g. using two sensors measuring the same value (Hwang et al., 2010). Analytical redundancy uses mathematical models. Model-based fault detection generates residuals, i.e. the differences between the observed values and the values calculated by a model (Hwang et al., 2010). From the residuals, faults are determined. The residuals are expected to be close to 0 if the system is in a fault-free state (Hwang et al., 2010).

Among the most important properties of a fault diagnostic system, (Venkatasubramanian et al., 2003c) identifies the capability to detect faults that were not modelled. Further important properties are identified, like high user confidence into the system's classification accuracy, an error estimate enabling to come up with a confidence level, and that the modelling effort should be as minimal as possible.

Following (Venkatasubramanian et al., 2003c), fault detection ranges from accurate modelling to not using any model but strictly relying on historical data. (Venkatasubramanian et al., 2003c) categorises fault detection into quantitative model-based methods, qualitative model-based methods, and process history based methods. While quantitative models are based on mathematical formulations, qualitative models use qualitative knowledge (Venkatasubramanian et al., 2003a) such as a knowledge base consisting of pre-defined if-else rules or a fault tree (Venkatasubramanian et al., 2003a). In (Venkatasubramanian et al., 2003b) fault detection and diagnosis techniques are discussed that rely on historical data. From the historical data, features are extracted. This corresponds to a machine learning approach.

In (Venkatasubramanian et al., 2003c) the conclusion is drawn that quantitative model-based methods are often not applicable in industrial applications due to the high dimensionality of the data, the system's complexity and non-linearity.

For techniques relying on historical data, the problem is identified that typically no data set with all faults is available that would allow to fully learn the fault behaviour. On the other hand, data sets with normal behaviour are usually available (Venkatasubramanian et al., 2003c).

The process of fault detection is sketched in (Venkatasubramanian et al., 2003c) as follows. The input variables are referred to as the measurement space, which is mapped to feature space using a priori knowledge (e.g. feature selection). This mapping follows the assumption that the input variables can be better classified in the selected feature space. Based on the feature space, some function is used to come up with decisions.

Fault detection is one part of fault diagnosis which comprises fault detection and fault isolation (Isermann, 2006). In (Isermann, 2006) two ways of fault diagnosis are distinguished, where the latter one corresponds to a machine learning approach:

1. theoretical modelling: creation of a model based on mathematical formulations

2. experimental modelling: learning from measurements, possibly starting with a priori knowledge

According to (Isermann, 2006), model-based fault diagnosis approaches are not feasible for highly complex applications, due to the models' complexity. In those cases data driven fault diagnosis can be utilised using measurements of a process in normal operation.

3.4.1 Anomaly detection techniques

Anomalies in time series can either be detected in the raw time series or based on features extractable from time series data. The raw time series can be pre-processed

in order to reduce the dimensionality (Han and Kamber, 2006) or transformed to an alternative representation (Keogh and Lin, 2005). Various approaches for anomaly detection that can be used for time series data, have been proposed in the literature. Some examples are listed here:

- User-driven techniques: Anomalies can be manually detected by domain experts supported by intelligent ways of presenting the data (Theissler et al., 2010).

- Instance-based techniques: Instance-based techniques like k-nearest neighbour (k-NN) can be used to classify unseen data to one of the classes normal or abnormal (Hodge and Austin, 2004). Classification is based on the distances to instances in a knowledge base. Either the entire time series or feature vectors can be stored in the k-NN knowledge base. An advantage is that the calculated distance can be used as an anomaly score.

- Rules or decision trees: Using Allen's interval logic (Allen, 1983), temporal rules can be discovered (Hoeppner, 2002) and a set of rules or a decision tree (Mitchell, 1997) be created. Instances not obeying the rules can be classified as abnormal.

- Statistical methods: Normal and abnormal data can be distinguished by modelling the probability distribution functions of the normal data and the faults.

- Clustering techniques: Clustering techniques like k-means can be used to find clusters of normal data (Hodge and Austin, 2004; Mitsa, 2010).

- Artificial neural networks: Artificial neural networks (ANNs) (Bishop, 1995) can either be used in feature space to distinguish between the normal and abnormal class or on the raw time series by predicting future values and evaluating the prediction error.

- Hidden Markov models: Hidden Markov models (Laxman and Sastry, 2006) can be used to determine whether the order of subsequences is normal.

- Limit checking (Isermann, 2006): In univariate time series faults can be detected by limit checking, i.e. based on minimum and maximum thresholds. More

advanced techniques monitor trend, mean value and variance. Limit checking can be improved using adaptive thresholds, i.e. adapting the threshold to the current operation mode.

- Signal models: Fault in periodic signals can be detected using Fourier analysis. For non-periodic signals, short-time Fourier transformation or wavelet transformation can be utilised (Isermann, 2006).

- ARMA models: ARMA models (Laxman and Sastry, 2006), also referred to as Box Jenkins-models, can be used to fit a model to either the entire time series or subsequences. Anomaly detection can then be done based on distances between the model parameters or on the probability that a certain model created a sequence. It was shown to work in (Deng et al., 1997), but is not viewed as a promising approach because the order and the coefficients of the models have to be determined.

- Application-specific models: Application-specific models describing the correct system behaviour can be used. The creation of the models can be complex depending on the application. Detection is based on the comparison between the behaviour of the process with the behaviour of the application-specific model. The difference is referred to as residuals. The underlying assumption is that a fault introduces a detectable change to these residuals (Isermann, 2006). Alternatively fault models can be used.

3.4.2 Discussion

From the discussion on fault and anomaly detection, the following two issues are identified as most important: (1) the capability to identify unknown or unmodelled faults and (2) the statement that fully modelling a process is infeasible for complex systems. This should guide the creation of an anomaly detection system throughout the subsequent chapters.

Two further distinctions between anomaly detection techniques are viewed as important for this Thesis: whether the detection technique can be used for online- or offline detection and if it outputs a class label or a continuous anomaly score.

It is a crucial difference whether the anomalies are to be detected in an online manner, or offline on previously recorded data. Online detection puts real-time constraints on the detection system. The detection must not be time consuming, which restricts the choice of the technique to a subset of the known anomaly detection approaches. If e.g. machine learning approaches are used, the training period may be computationally expensive but not the detection process. So for example a decision tree could be learnt using a training set. During evaluation of unseen data instances, only a few simple tests need to be done on the data instance in order to determine the branches of the learnt decision tree.

This Thesis focuses on the evaluation of previously recorded time series. Therefore, it focuses on offline anomaly detection, i.e. there are no real-time constraints on the detection process. This means that anomaly detection techniques, where the detection process is computationally expensive, may be taken into account.

Anomaly detection techniques differ in the type of the provided output. One type of output is a label stating whether a data instance is normal or abnormal. Looking at two detected anomalies, no statement can be made in terms of which one is more relevant.

In contrast to providing just one of two labels, some techniques return an *anomaly score*. The anomaly score is a figure to rank the anomaly, which can be a distance, a confidence value or an application-specific figure. Normal and abnormal data can then be distinguished by introducing a threshold.

Having an anomaly score attached to each data instance is particularly advantageous when many anomalies are expected to be reported. Based on the anomaly score the presented anomalies can be ordered and thereby prioritised. The number of results can easily be reduced by simply altering the threshold – adaptively or user-defined. Therefore an anomaly score is desirable.

3.5 Conclusion

Finding anomalies can be done in a user-driven, interactive way, which in this Thesis is referred to as interactive anomaly detection. The user manually detects anomalies based on knowledge and experience using an intelligent computer system to support the investigation. The user is required to have a deep understanding of the time series to be investigated, i.e. she/he has to be a domain expert and has to be familiar with the detection system.

In contrast, semi-autonomous or autonomous approaches work by either manually pre-configuring a knowledge base or by learning from historical data. When learning from historical data, as a first step, knowledge is extracted from a training data set. The goal is to extract knowledge in a way to be able to classify unseen data. Subsequently a test period is conducted using a test data set containing instances with known class labels to evaluate the classification results. The training and test period may be repeated several times in order to optimise parameters or the constitution of the training data set. Based on the extracted knowledge, unseen data is classified as normal or abnormal in the performance period.

Summarising, approaches to detect anomalies can be categorised by the level of user interaction required during anomaly detection:

1. *interactive anomaly detection*: The user manually detects the anomalies based on her/his domain knowledge and experience, supported by an intelligent computer system.

2. *semi-autonomous anomaly detection*: A system suggests anomalies, the results are evaluated by a domain expert during the entire operating time.

3. *autonomous anomaly detection*: After initial learning or configuration, a system fully autonomously classifies instances as normal or abnormal.

In the next chapter, interactive anomaly detection utilising visual data mining techniques is introduced. Following that, autonomous anomaly detection is investigated in Chapter 5 and an approach is proposed in Chapter 6 and Chapter 7.

Chapter 4

Interactive anomaly detection

This chapter introduces visual data mining as a user-driven approach for anomaly detection. By conducting case studies with real-world data, it is shown how the analysis of test drive recordings can be improved. Conclusively, the shortcomings of a user-driven approach are discussed.

In this chapter, visual data mining techniques are surveyed and an adapted approach for the detection of anomalies in time series is proposed. The user-driven analysis of recordings from vehicle tests and long-term traffic measurements is shown in case studies. Some of the results were published by this Thesis' author in (Theissler et al., 2010) and (Theissler et al., 2011).

4.1 Surveying visual data mining

In a survey paper on visual data mining (Ferreira and Levkowitz, 2003) the authors state that the presence of temporal dimensions in the data is a determinant factor for the appropriate visual representation technique. Proper visualisation techniques can

support the user during interactive exploration of the data (visual data exploration). Furthermore, visualisation can be used for classical data mining tasks like cluster detection, classification or pattern discovery, and hence can be used for anomaly detection.

Numerous different techniques are known starting with standard 2D- or 3D-plots or charts as known from calculation tools to geometric techniques or iconic displays. In (Keim, 1997) Keim gives a variety of examples: Geometric techniques are concerned with multidimensional data, data that cannot be visualised using e.g. standard x/y plots. The landscape technique for example visualises the data as a perspective landscape. The parallel coordinates technique (Few, 2006) maps the dimensions of the data to axes in a chart. Further techniques are scatter plot matrices relating time series pairwisely.

Following the definition of Fayyad (Fayyad et al., 1996), knowledge discovery is the "nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". This can for example be done in an automatic manner by applying data mining algorithms (Han and Kamber, 2006), or using application-specific models, or in an interactive way by integrating the user into the process. In this chapter the latter one is described and referred to as interactive anomaly detection.

The term information-seeking mantra, which could be described as the steps to be taken to discover knowledge (Fayyad et al., 1996) from visualised data in general, was introduced by Shneiderman in (Shneiderman, 1996). Keim related this principle to visual data exploration in (Keim, 2001), where the exploration process takes place in an interactive way.

In the survey paper (Ferreira and Levkowitz, 2003), the evolution from visual data exploration to visual data mining is discussed. The more recent term visual analytics was introduced in (Thomas and Cook, 2005), which is defined as the science of analytical reasoning facilitated by interactive visual interfaces.

Although there is a variety of visualisation techniques, meaningful visualisation techniques for multivariate time series data with many variables are rare. Most of the current visualisation techniques were not especially developed to be applied on time

series data or use only a few time-dependent variables. For the analysis of multivariate time series, enhanced visualisation techniques are required.

4.2 Enhancing visual data mining for anomaly detection

The classical way to visualise time series data is to plot line graphs, where the values are plotted as a function of time. If several data items are to be visualised, this typically leads to overplotting and relating several variables is not possible. In recordings from test drives there is a variety of different time series that have to be related.

In order to detect anomalies in an interactive manner, techniques from the field of visual data exploration (Ferreira and Levkowitz, 2003) and temporal data mining (Antunes and Oliveira, 2001; Mitsa, 2010) were identified to be most promising. Existing techniques were adapted and enhanced in order to cope with multivariate time series data, so that the data can be visually related or queried. For example outliers within one of the univariate time series, deviations in the relationship of multiple dependent time series, or unexpected correlations between independent univariate time series can be detected.

In this chapter an interactive approach is proposed that is designed to work on time series data. The approach combines the two existing visual data mining techniques "parallel coordinates" (Inselberg, 1985) and "scatter plot matrix"(Keim, 2002). The two techniques are enhanced to cope with multivariate time series data and to enable the user to formulate sophisticated filtering and querying operations. This is combined with a time series query tool that allows to graphically formulate a search pattern that is searched in the recordings based on shape-based distance measures. Additionally, a variety of techniques to pre-process and transform the input time series are integrated.

4.2.1 Relating time series pairwise: Enhancing scatter plot matrices

Visually relating two variables can be done using a scatter plot, where each attribute is mapped to one Cartesian axis. For several variables a so called scatter plot matrix can be used, which consists of a matrix of pairwise projections of the attributes as shown in Figure 4.1.

Information about time is not contained in those individual x/y-diagrams. Visualising several time series using scatter plot matrices will therefore enable the user to detect anomalies in the pair-wise dependencies between signals, but without being able to draw conclusions about the point in time.

The idea of a scatter plot matrix was enhanced in this Thesis by a number of features: data points in one of the x/y-plots can be highlighted by the user by drawing a rectangle in one of the x/y-plots. Each data point is identified by its timestamp. Based on the timestamp, the highlighting is propagated to the values of all time series in the remaining x/y-plots. Sophisticated queries can be formulated incrementally by allowing subsequent operations to be linked by the Boolean operators AND, OR and NAND which allows to interactively search for interesting parts in the data by refining or extending the currently highlighted data points. If a high number of data points is highlighted, it is beneficial to highlight the data points using a colour gradient, in order to distinguish the data points.

Depending on the number of data points and size of the display up to fifteen different attributes can be related that way. The relation though is strictly done pairwise. However, in recordings from a vehicle, one signal is often dependent on a combination of several variables.

4.2.2 Relating n time series: Enhancing parallel coordinates

Relating n variables can be done using parallel coordinates, introduced by Inselberg in (Inselberg, 1985). In this Thesis, an enhancement of parallel coordinates is proposed

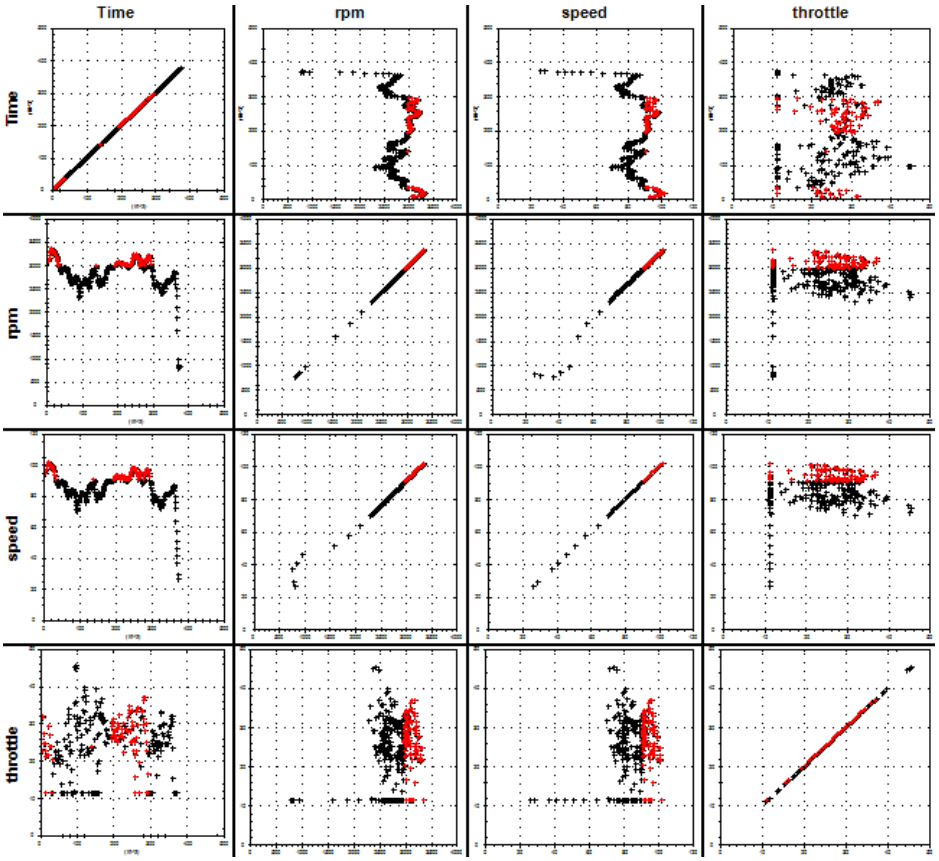


Figure 4.1: Scatter plot matrix relating the signals engine rpm, vehicle speed, and throttle position from a test drive. All data points where the vehicle speed was greater than 90 km/h are highlighted.

to cope with the time series data from recordings of test drives. Relating n time series of equal length d is done by drawing $n+1$ parallel axes. The time stamps are mapped to the first axis and each time series is mapped to one further axis. The values are mapped to the corresponding position on the axes, starting with low values at the bottom of the screen, and then connected by line segments as shown in Figure 4.2.

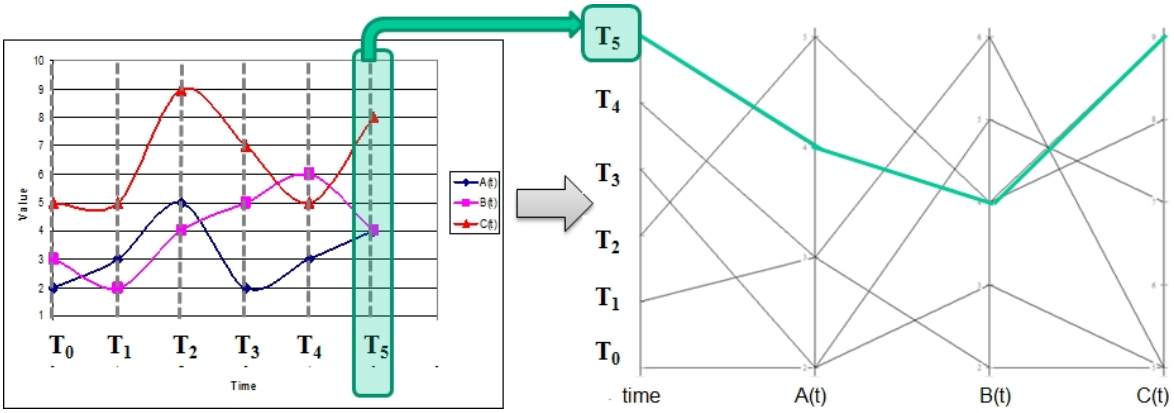


Figure 4.2: Mapping of multivariate time series with 3 signals and 5 time stamps $T_1 \ldots T_5$ to parallel coordinates.

A parallel coordinates plot of a large data set leads to a visualisation that does not reveal useful information due to massive overplotting. One way to cope with this is to draw transparent lines. This is done using the α-channel of the colour (Wegman, 2003). Ranges with a high density of data items get coloured in a more intense way while the remaining parts appear transparent. Data distribution, general dependencies, and outliers can be spotted this way as can be seen in Figure 4.3.

In addition, colouring the data items of the parallel coordinates plot using a colour gradient (Wegman, 2003) w.r.t. a selected dimension leads to a visualisation that allows the overall-structure of the data to be recognised. The level of transparency as well as the axis the colour gradient is based on, can be interactively influenced by the user.

A value range can be selected and thereby highlighted by the user (Few, 2006), referred to as brushing. Individual brushing operations work on selected axes. In this Thesis, the enhancement was made that consecutive brushing operations can be linked by Boolean operators. The Boolean operators are applied to the set of currently brushed

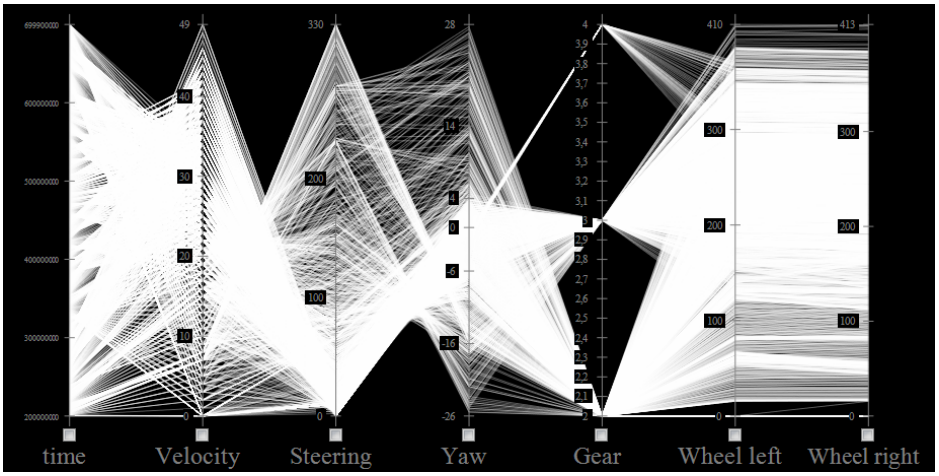


Figure 4.3: Parallel coordinates with transparent items showing which value ranges were dominant during one test drive.

data items and the data items contained in the current brushing operation. This way, sophisticated queries can be formulated on a time series data base. For example in Figure 4.4, all data points where the vehicle was in the 3rd gear while going between 40 and 50 km/h are shown. The query was formulated as follows:

1. the velocity signal is in the range of 40 . . . 50 km/h

2. AND the vehicle is in the 3rd gear

The highlighted data items can be coloured based on a colour gradient in order to distinguish the individual data items. In order to focus on the relevant parts of the data, brushed or unbrushed data items can be removed.

As opposed to visual analytics tools for the analysis of unstructured data, working on time series allows to integrate functions defined for time series like differentiation, integration, point-wise arithmetic, and time series distance measures (Moerchen, 2006).

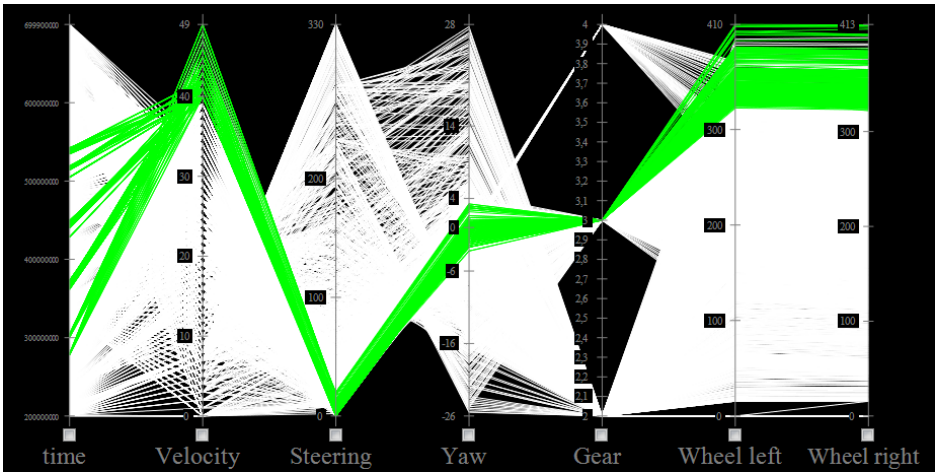


Figure 4.4: Parallel coordinates with Boolean brushing operations applied to highlight subsequences where the vehicle's velocity is in the range of 40 ... 50 km/h and the vehicle is in the 3rd gear.

4.2.3 Time series pattern query: Searching for patterns in univariate time series

An approach, inspired by (Wattenberg, 2001), was developed that allows to query a time series data base for a user-specified search pattern. The search pattern can be formulated graphically as can be seen in the left panel at the bottom in Figure 4.5. The found matches are marked in the input time series as shown in the top panel in Figure 4.5.

The search pattern is moved over the input time series and the distances for each time point are calculated. The two common distance measures Euclidean distance (Mitsa, 2010) and dynamic time warping (Mitsa, 2010) are used. The Euclidean distance is calculated point-wise and summed up as given in eq. (4.1). Small misalignments on the time axis, e.g. a stretched pattern, lead to a big distance. The way the Euclidean distance is calculated between a search pattern and an input time series is graphically shown in Figure 4.6.

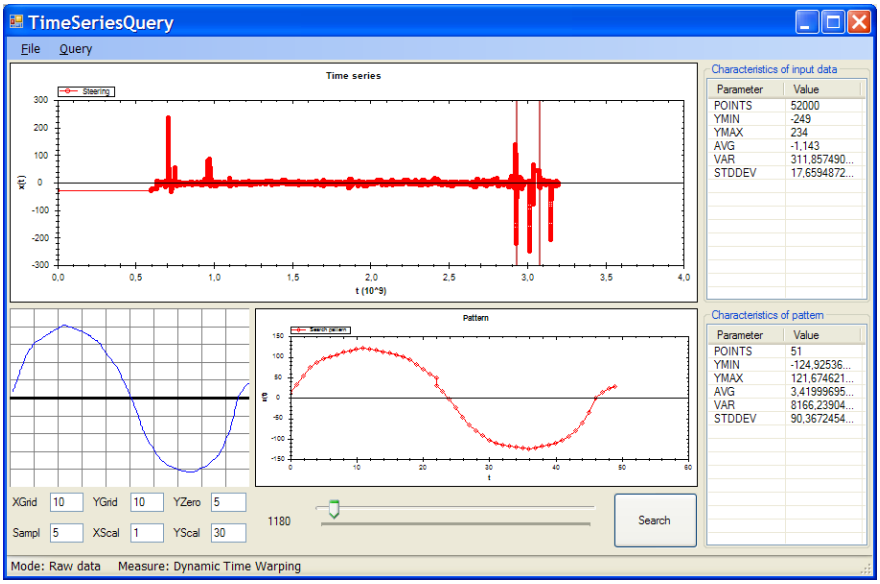


Figure 4.5: Graphically formulating a query for a specific driving manoeuvre where a right curve is followed by a left curve.

$$D(X_{T1}, X_{T2}) = \sqrt{\sum_{i=1}^{n} (X_{T1_{t_i}} - X_{T2_{t_i}})^2} \tag{4.1}$$

To assign smaller distances to patterns that are similar but differ in length, dynamic time warping (DTW) is used, where the distance is not calculated strictly pairwise, but neighbouring data points are considered as well. The calculation of the distance is visualised in Figure 4.7(a). The DTW distance is the smallest distance between two time series taking into account misalignments. It is determined by the so-called warping path illustrated in Figure 4.7(b).

If not the absolute values of the search patterns are relevant, but rather the shape, the time series can be searched for multiple scaled versions of the search pattern. In (Furnas and Buja, 1994) the authors propose to use a scaling factor l and create variations of the given time series in the interval $[l, \frac{1}{l}]$. Additionally, the input time series can be transformed to an alternative time series representation – e.g. the symbolic representation SAX (Keogh et al., 2006), which allows to develop faster search algorithms.

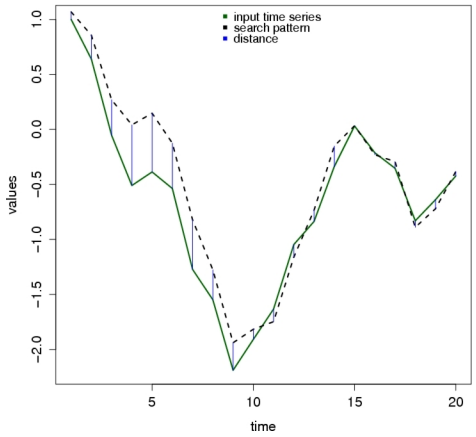
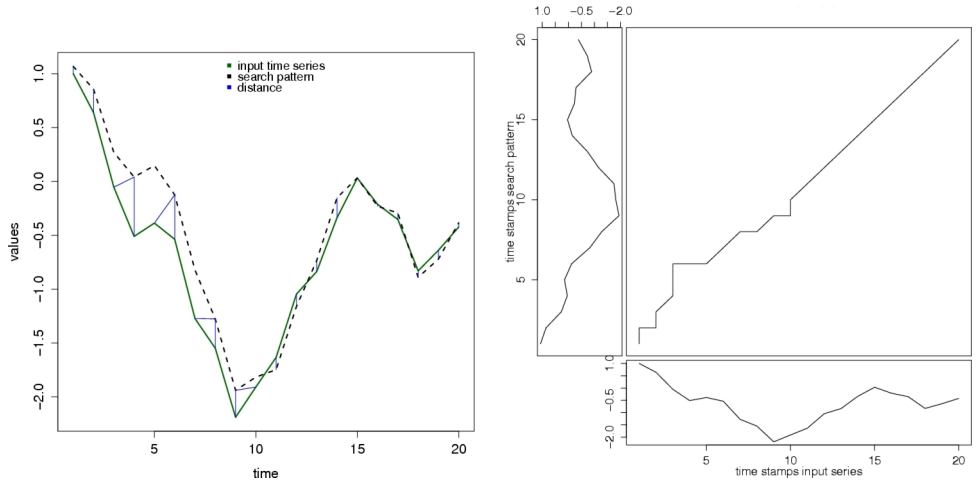


Figure 4.6: Euclidean distance between a search pattern (black, dashed line) and an input time series (green, solid line).



(a) Distance between a search pattern (black, dashed line) and an input time series (green, solid line) calculated with DTW.

(b) The warping path to calculate the DTW distance.

Figure 4.7: Distance between a search pattern and an input time series calculated with the dynamic time warping distance measure.

4.2.4 Interaction between the techniques

The key to the interpretation of the recordings is linking the techniques and propagating selected data items between the visualisations as illustrated in Figure 4.8.

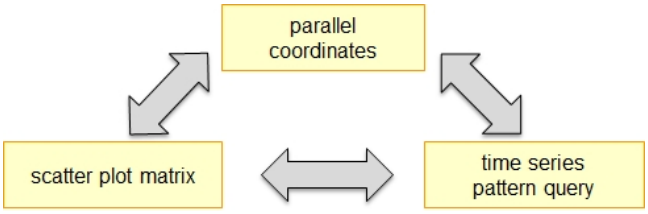


Figure 4.8: Interaction between the used visual data mining techniques.

The combination of the parallel coordinates plot with a scatter plot matrix offers functionality like the identification of outliers in the scatter plot matrix and the propagation of the highlighting to the parallel coordinates plot or vice versa. This way, the benefits of both techniques can be used.

The additional integration of the tool to graphically query a time series for a search pattern offers the additional benefit of being able to formulate very specific shape-based search queries. The search results can be highlighted in the parallel coordinates or scatter plot matrix and can thereby be refined by applying the Boolean brushing operations, or the results can be related to further signals.

4.3 Experimental results on real data sets

The section discusses how the introduced visual data mining techniques can be used for the detection of anomalies. To investigate the effectiveness of the approach, experimental results are shown on data recorded during test drives with a vehicle, data from a HiL system, and on traffic measurements.

4.3.1 Case studies based on recordings from an in-vehicle network

In this section, experimental results obtained from applying the visual data mining techniques to real-world measurements from in-vehicle networks are presented. It is shown, how the visual data mining techniques can be used for detailed analysis of one or multiple recordings.

4.3.1.1 Description of the data sets

In the remainder of this section, recordings from an in-vehicle network are analysed. The data was recorded from the CAN-bus (see Section 1.2) during test drives with a vehicle or from a HiL-system. In order to obtain a common time base and equidistant time series, the data was resampled prior to the analysis. Approximately 10 minutes of a recording from a test drive are shown in Figure 4.9, where a pre-selection of 6 signals is plotted. The plot shows the vehicle's velocity, absolute value of the steer angle, yaw rate, gear, and wheel speeds of the rear wheels.

4.3.1.2 Case study: Querying a recording of a test drive

A recording from a vehicle with automatic transmission is investigated. As part of the user-driven fault detection, all subsequences in the recording shown in Figure 4.9 are searched, where the gear is changed while the vehicle is going through a curve.

From the recording, the relevant signals are extracted and visualised using the enhanced parallel coordinates, where each signal results in one axis. Utilising the highlighting mechanism with Boolean operators yields the result set after four steps. The following query was iteratively formulated:

1. copying and derivation of the signal holding the gear

2. selection of the entire range of gears

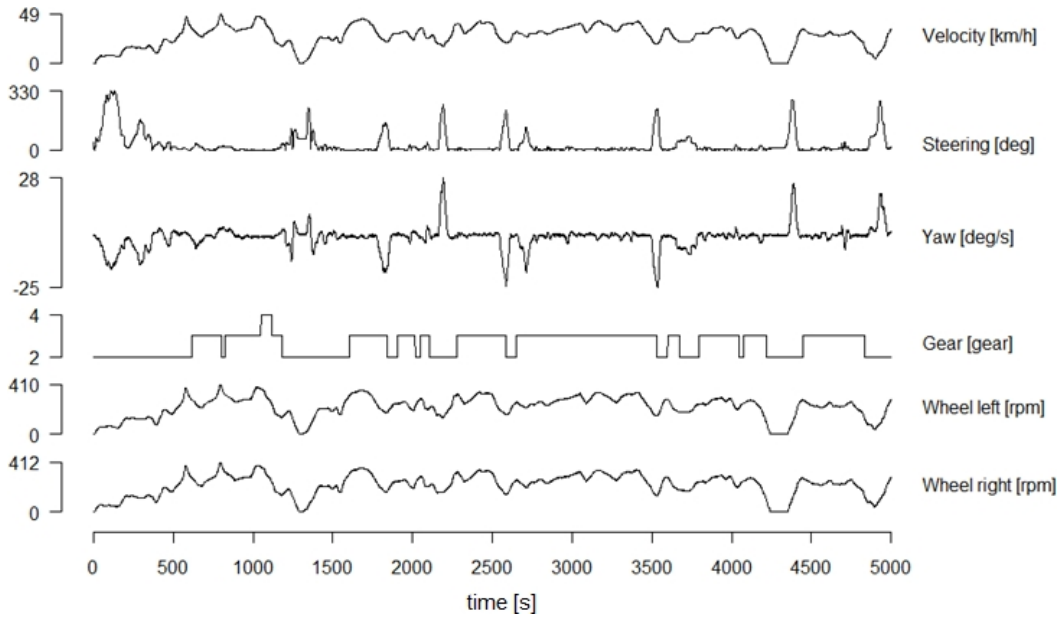


Figure 4.9: Recordings from an in-vehicle network from a test drive.

3. refining of the selection to all values where $gear_{derived} \stackrel{!}{=} 0$

4. further refining of the selection to all values where the vehicle is going through a curve, i.e. where the absolute value of the steer angle is $\gg 0$

The search results are highlighted in Figure 4.10. From the first axis it can now be seen that the requested search pattern occurred four times and it was only geared down from third to second gear. In addition, the difference of the wheel speeds between left and right rear wheel, while the gear was shifted can be seen from the axes "wheel left" and "wheel right".

To keep this example simple, a recording of just 10 minutes was used. When applying this query mechanism to recordings of several hours it becomes especially powerful for user-driven data analysis.

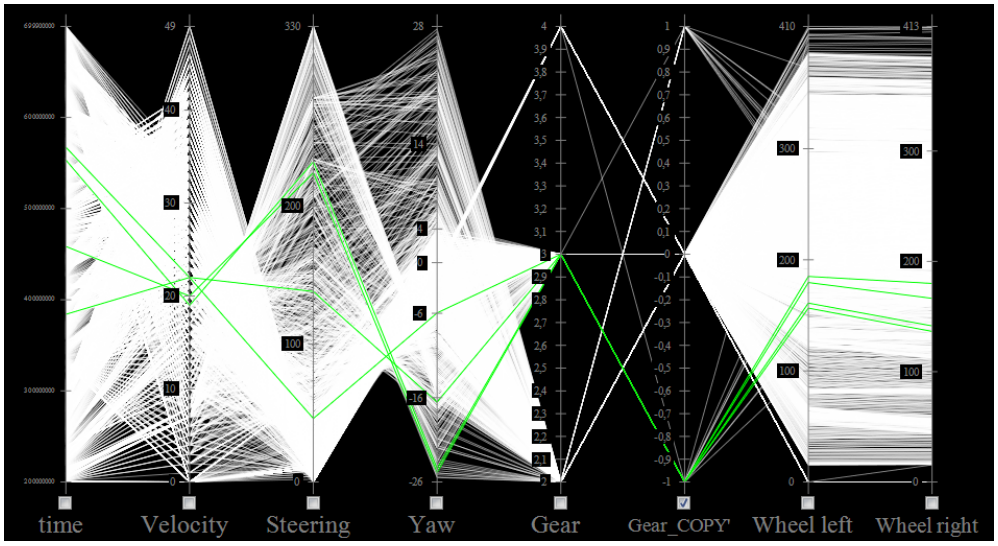


Figure 4.10: Parallel coordinates highlighting all changes of the gear, where the vehicle was going through a curve.

4.3.1.3 Case study: Timing analysis of recordings from an in-vehicle network

Using parallel coordinates, a new way is proposed to present the timing behaviour of the entire in-vehicle network at one glance.

The ECUs inside the vehicle communicate by sending messages on the bus system. The majority of messages on the CAN bus is sent in a cyclic manner, following a pre-defined cycle time. Real-time operating systems are running on the ECUs with scheduled tasks reading signals, calculating results and sending the results on the bus. Failure to meet the given timing requirements can result in failure of a vehicle function.

The CAN bus (Mayer, 2010a) is not deterministic, the message priority on the bus is based on the message id. A systematic jitter is observable for cyclic messages.

In order for a vehicle function to work properly, the properties of the operating system task, the wall clock time of the algorithm running in the task and the message priority on the bus system need to be adjusted. An improper constellation of these parameters is one reason for erroneous deviations from a pre-defined cycle time. The deviation

could occur for certain operating points of an ECU, if for example an algorithm's wall clock time exceeds a pre-defined tolerance for certain input values.

The data for this experiment was recorded from a HiL-system. For each message the time stamp, message id, cycle time, and length of the payload is available. Based on this data further attributes are deduced in a pre-processing step. The following data items are deduced and plotted as shown in Figure 4.11.

1. time stamps of messages
2. time span between two consecutive messages
3. time span between two messages with same id
4. deviation from cycle time in μs
5. deviation from cycle time in %
6. cycle time
7. CAN id
8. length of payload (DLC)

Timing deviations for all messages can be detected by highlighting deviations from cycle time by more than 10% with a colour gradient (green, yellow, red), where the deviations marked in red are most critical (see Figure 4.11). For further analysis, the message ids and the point in time can be deduced from the highlighted fraction of the data. As a consequence, it is now possible to see if timing violations occurred for specific message ids, for certain lengths of the payload or during a certain time span.

Further analysis showed that for one of the messages the cycle time was incorrectly configured at the HiL test stand. In addition there was a burst of timing violations at the end of the recording, which was caused by shutting down the test run while still recording.

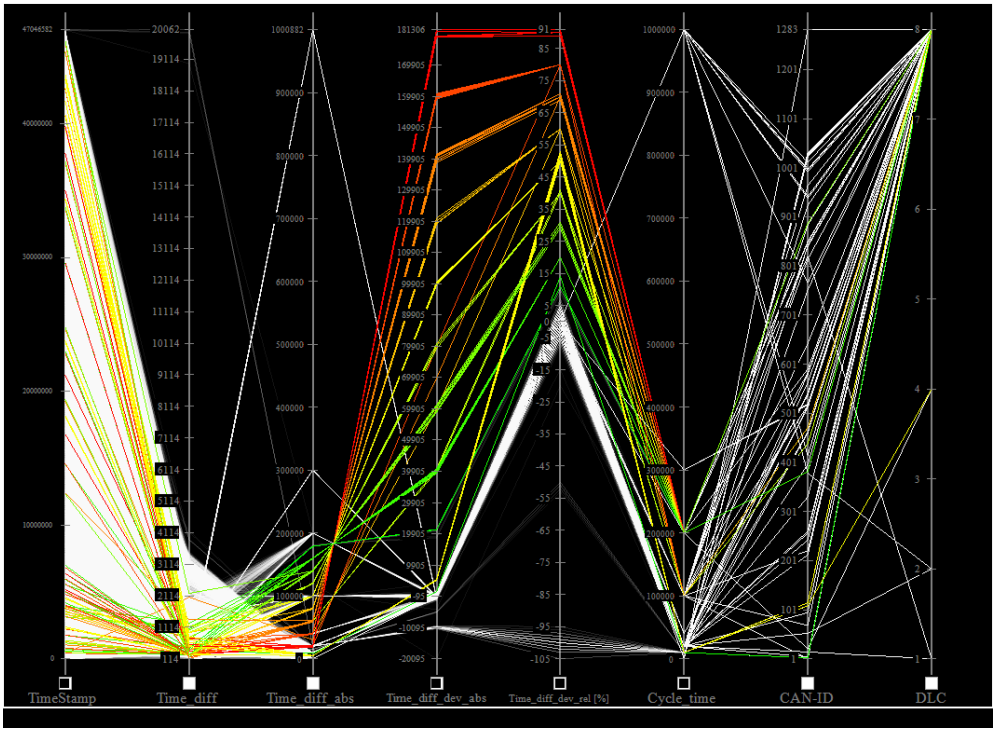


Figure 4.11: Timing analysis of in-vehicle network traffic recorded from a HiL test stand. The time stamps, data length, and can identifier of messages violating the cycle time can be identified.

4.3.1.4 Case study: Detecting abnormal driving situations throughout various test drives

The previous case study worked on one individual recording. A challenge often encountered is the need to include various recordings into the analysis. In this case study, the recordings of four test drives are included. The four recordings are imported and an additional axis holding the number of the recording is generated. In order to get a quick overview, the data items are coloured following a colour gradient [blue, green, yellow, red], based on the axis holding the steer angle.

The relation between the steer angle and the vehicle's yaw is usually anti-proportional, in other words the direction of the vehicle follows the steering wheel angle. When the vehicle is going through a left curve, the steering wheel angle is a negative integer, the yaw in turn is a positive integer. A deviation from this relation can be detected following some of the red lines connecting high values on the axes holding the steer angle and the axis holding the yaw signal, marked in Figure 4.12.

In order to focus on this anomaly, the data items are selected using the subsequent brushing operations:

1. steering wheel angle right
2. refining selection to values where the yaw rate indicates that the vehicle was turning left
3. further refinement by excluding the data points where the vehicle acceleration is close to 0

From the resulting visualisation shown in Figure 4.13 it can be seen that the searched constellation occured in two of the test drives. One of the test drives was conducted on icy road conditions, so the vehicle was sliding at that point in time. The second occurrence points to a situation where a steering maneouvre to the left was abruptly followed by a steering maneouvre to the right.

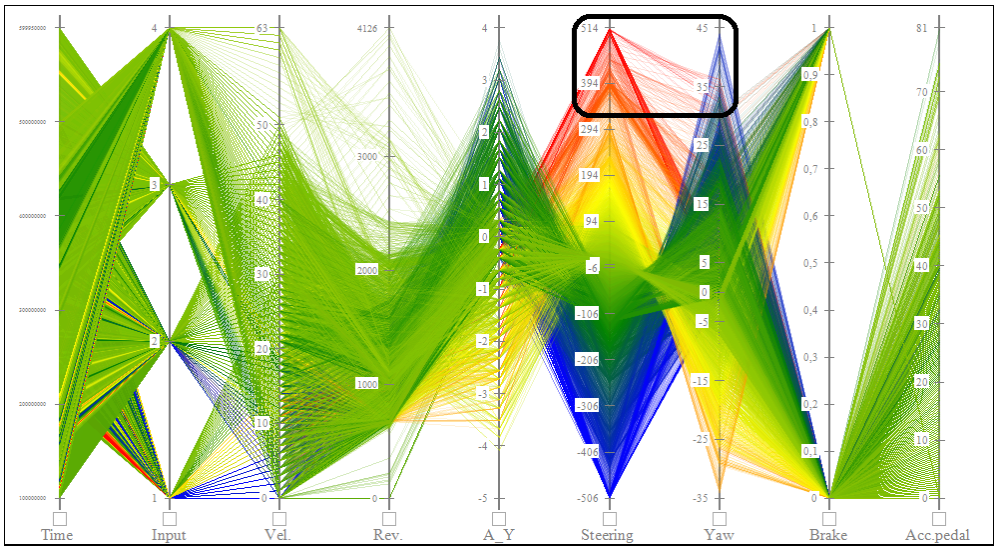


Figure 4.12: Parallel coordinates with a colour gradient showing the structure of the data. An abnormal deviation of the anti-proportional dependency between the steering wheel angle and the yaw signal can be detected.

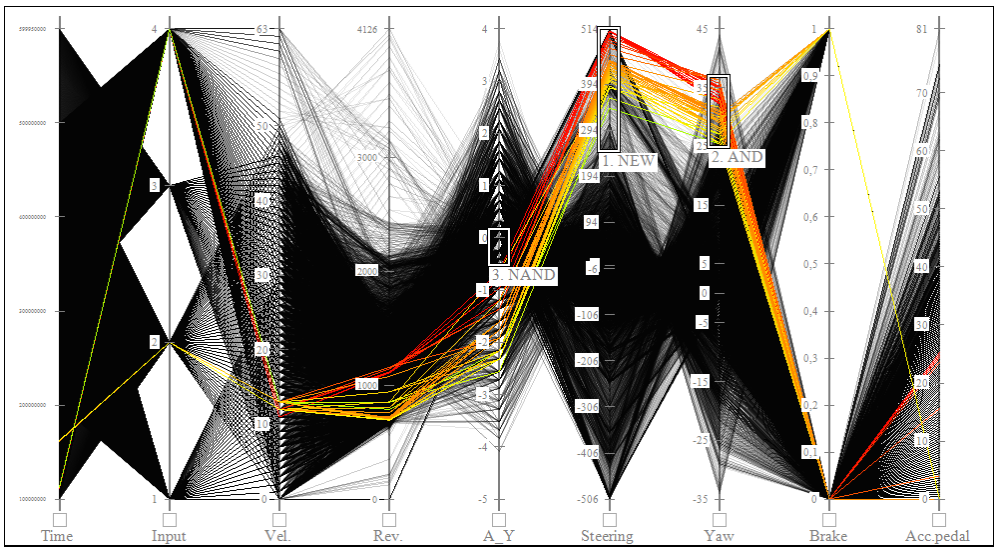


Figure 4.13: Isolating abnormal driving situations by querying using Boolean operators.

4.3.1.5 Case study: Querying for specific driving manoeuvres

If specific patterns are known, e.g. profiles of one of the signals, the recordings can be queried as shown in this case study. In order to find all occurrences of right curves followed by left curves, the user graphically defines a search pattern based on the signal holding the steering wheel angle.

The distance between the search pattern and the steer angle signal is calculated for each data point, which results in a distance vector. The user configures the allowable deviation from the search pattern by adjusting a threshold. Two occurrences were detected and are marked as matches in Figure 4.14.

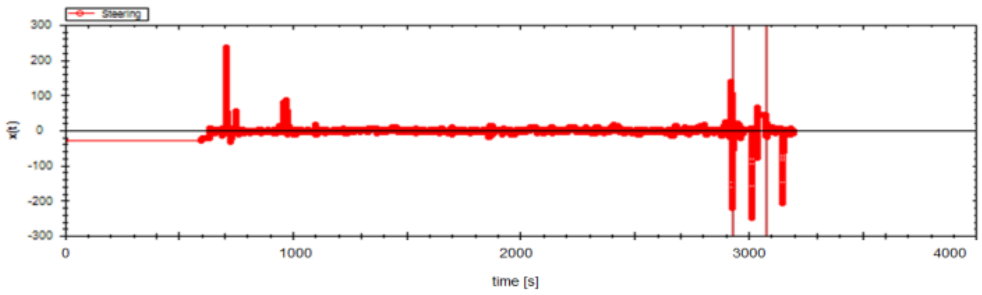


Figure 4.14: Search results for right curves followed by left curves.

4.3.1.6 Case study: Anomalies in dependent signals

In Figure 4.15 the vehicle's velocity, revolutions, steering wheel angle and the wheel speed left and right are put into relation. From the visualisation it becomes obvious that the speed of the right wheel was stuck at 255 for a short period of time marked with a circle. This error was manually injected into the data set and could have been caused by an erroneous sensor or a software fault.

Figure 4.15: Dependency between speed of left and right wheel is violated for a short period of time (marked red).

4.3.2 Case studies on long-term traffic measurements

The case studies in this section are based on real-world traffic data. The data set is higher-dimensional than the ones used in Section 4.3.1.

The following will be shown: (1) the detection of errors in the underlying detector network, (2) the identification of problem sections on a motorway based on long-term measurements, and (3) the detection of abnormal traffic situations.

Detection of anomalies in the traffic measurements can have several goals: either to investigate atypical traffic behaviour in a goal-oriented manner or to remove the

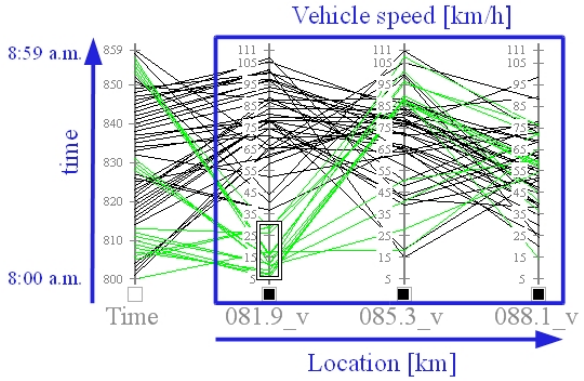


Figure 4.16: Enhanced parallel coordinates showing one hour and three detectors

anomalies in order not to consider them in the deduction of general statements about the traffic behaviour for a given road section.

4.3.2.1 Description of the data set

The data was recorded between November 2009 and January 2010 by 33 stationary loop-detectors positioned non-equidistantly at a 40 km long section of a German motorway. The detectors measure the velocities and quantities of passing vehicles in one direction of the motorway 24 hours a day with a sample rate of one minute. The vehicles' velocities were used resulting in approximately 4 million data points.

4.3.2.2 Introductory example

In Figure 4.16 a small subset of the data is visualised using the enhanced parallel coordinates. The velocity values of three of the 33 detectors are visualised over a period of one hour. Additional information was added in dark blue indicating how the time, the location, and the vehicle speed are illustrated in the plot.

The first vertical axis in Figure 4.16 shows the timestamps, 8.00 a.m.-8.59 a.m. of one pre-selected day in this example. Each of the remaining vertical axes shows the velocities measured by one detector in that period. The location of the individual

detectors is shown in horizontal direction. The axes holding the velocities are ordered in a way that a vehicle would pass the detectors from the most left axis to the most right axis. Each of the axes is labelled according to the detector's position, so for example the axis labelled '081.9_v' shows the velocities measured by the detector at motorway position 81.9 km.

Each line connecting the axes from left to right corresponds to the vehicles' velocities at one point in time. The three axes holding the velocities are aligned to have a common minimum and maximum value, i.e. one horizontal line connecting the axes would correspond to equal velocities at all detectors. In order to be able to distinguish the temporal and spatial information, the detector axes are indicated by a black square at the bottom of the axis. Axes without a black square hold the temporal information. Due to the fact that the detectors output one velocity value per minute and one hour is contemplated, there are 60 lines connecting the axes in this example.

The benefit of the parallel coordinates plot is the possibility to easily formulate queries by highlighting selected data items. In Figure 4.16, the data set was queried to show velocities below 30 km/h at the detector at 81.9 km, indicated by the rectangle on the bottom of the second axis. The brushed data items are coloured in green. Following the green lines, two statements can be deduced in this example plot: (1) at which points in time the low velocities occurred and (2) what the velocities were at that time at the remaining two detector positions.

4.3.2.3 Mapping of raw data to the visualisation

The mapping of the data to the vertical axes in the enhanced parallel coordinates visualisation is crucial and can be done in various ways. A first approach could be to plot three axes: one axis holding time stamps, a second axis holding the detectors, i.e. the spatial information, and a third axis holding the velocities. Due to the high data volume mapped to only three axes, this results in heavy overplotting. Therefore, the temporal information is plotted in vertical and the spatial information in horizontal direction as one axis per detector as shown in Figure 4.16.

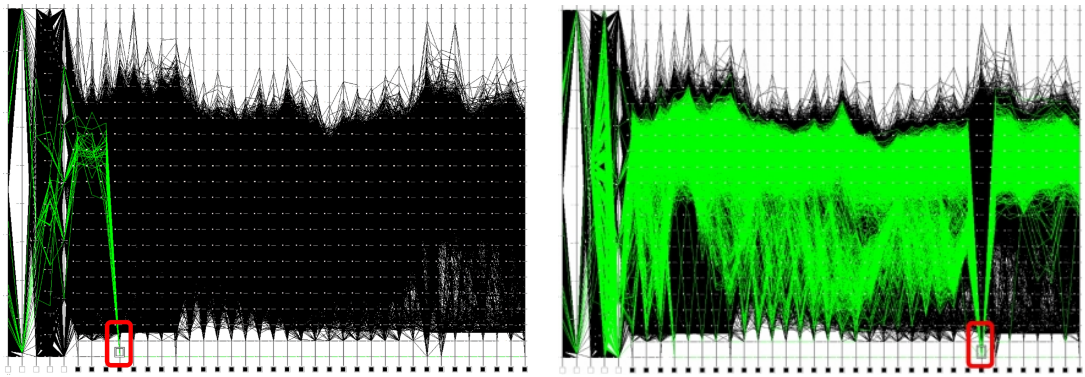
The time span between 6 a.m.-10 p.m. was chosen to be visualised, resulting in approximately 2.8 million data points. In addition, as opposed to plotting the raw time stamps, the temporal information was pre-processed, allowing for more goal-oriented queries. Mapping of the data to the vertical axes is done as follows (left to right):

1. index of data set
2. year and month
3. day
4. time of day (6 a.m.-10 p.m.)
5. weekday (0 = Sunday, 1 = Monday, ..., 6 = Saturday)
6. 32 axes corresponding to 32 detectors in the range of 81.9 - 120.4 km. The axes' positions on the screen from left to right correspond to the detectors' positions on the motorway in ascending order.

4.3.2.4 Case study: Detecting errors in the detector network

Relying on measured data from detectors, the evaluation of the underlying detector network is essential. In a pre-processing step, invalid or unavailable detector values were replaced with an error value of -10 km/h, the error value was chosen arbitrarily.

Highlighting the invalid values of one detector in Figure 4.17(a) shows that invalid values occur very sporadically for this detector. In Figure 4.17(b) the invalid values of a different detector were highlighted. A high number of error values becomes obvious at the axis of the detector at position 117 km. The erroneous detector values were highlighted using the brushing operation. The number of erroneous values can be obtained this way: 2223 erroneous values were detected, corresponding to 3.9% or an overall of 37 hours over the contemplated period of three months. As can be seen from the plot, other detectors show abnormal values as well, but to a much lesser extent.



(a) Sporadic invalid values for one detector. (b) Frequent invalid values for multiple detectors.

Figure 4.17: Evaluation of detector network by querying for abnormal values on two detectors.

The erroneous detector values in Figure 4.17 correspond to a "subsequence anomaly in a univariate time series" (see Section 3.3).

An evaluation whether the measured data of just one specific detector or a collection of detectors is erroneous at a given point in time is also possible, allowing to deduce conclusions about the cause of the error.

4.3.2.5 Case study: Identifying bottlenecks in a road-network

This case study has the goal to identify bottlenecks in a road-network. After the removal of one unreliable detector, the aim was the identification of problem sections on the motorway based on the time series data of the remaining 32 detectors.

The data set was plotted using transparent lines. Light grey lines correspond to individual values, while dark regions express the majority of values. Interactively adjusting the α-channel of the lines and thereby adjusting their transparency results in the visualisation shown in Figure 4.18.

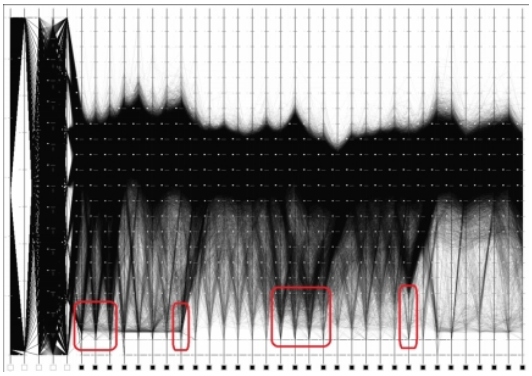


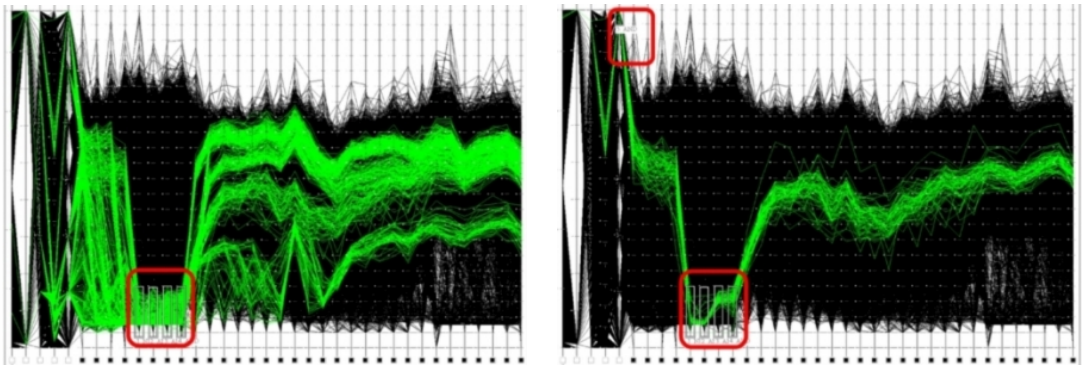
Figure 4.18: Identification of problem sections on a motorway.

Assuming that low velocities on a motorway indicate problems like traffic jams, valuable information can be deduced from Figure 4.18. Based on the contemplated three months, the following problem sections were identified for the motorway (marked by four red rectangles): 81 - 88 km, 98.8 km, 107 - 109 km, 115 km.

4.3.2.6 Case study: Detecting abnormal traffic situations

Investigating the reasons for traffic jams at specific locations requires to first identify where and when they were observed. Hence, goal-oriented search for specific traffic situations is beneficial. With the help of the introduced Boolean brushing operations, queries for traffic jams at specific sections can easily be formulated. In Figure 4.19(a) all traffic jams between position 95.4 km and 98.8 km were highlighted, by incrementally querying the detectors' axes for velocities ≤ 30 km/h. The resulting visualisation in Figure 4.19(a) allows (1) to evaluate the frequency of traffic jams at that subsection and (2) to obtain their points in time.

Contemplating the axis holding the weekday in Figure Figure 4.19(a) reveals the following: a small portion of the highlighted traffic jams occurred on a Saturday, which is unexpected. Enhancing the query in order to only highlight the subset of jams that occurred on a Saturday, results in Figure 4.19(b).



(a) Query for traffic jams at specific motorway subsection.

(b) Refined query for abnormal jams on Saturdays.

Figure 4.19: Queries for traffic jams.

From the plot two occurrences can be detected, both on Saturday 30th January 2010 (1) between 3.03 pm and 4.44 pm and (2) at 9.57 pm. The second occurrence was only one minute long and is therefore viewed as an irrelevant outlier. Research on the internet revealed the reason for the low velocities. On the 30th January 2010, there were dramatic road conditions in Germany due to snow and ice.

The detected anomaly integrates several variables and is a "contextual anomaly in multivariate time series" introduced as type 3 in Section 3.3.

4.4 Conclusion

In this chapter, visual data mining techniques were enhanced and applied to time series data from the automotive industry. Parallel coordinates, scatter plot matrices and a query mechanism for univariate time series were utilised.

With numerous experiments it was shown that the techniques are beneficial and allow for user-driven anomaly detection. **Aim 2** of this Thesis, decreasing the time needed for manual analysis of test drive recordings (see Section 1.4), was reached.

Detection accuracies cannot be generally measured as the approach relies on expert-knowledge. The user needs to have an understanding of the test drive data sets and needs to be familiar with the visualisation techniques. The approach supports the expert, but eventually relies on the expert to detect anomalies.

A problem that is inherent in all user-driven techniques is that the data analysis becomes infeasible for a high number of recordings due to the time an expert can spend for the analysis. An approach autonomously reporting anomalies would not be limited by the availability of an expert. The autonomous detection of anomalies using classifiers that are trained on training data sets will be addressed in the next chapter.

Summarising, the visual data mining techniques introduced in this chapter are vital for this research for following tasks:

1. user-driven anomaly detection as shown by the case studies
2. data reduction to focus on specific scenarios in test drives
3. selection of a training data set for autonomous classifiers
4. analysis of the results of an autonomous detection approach, leading the user from reported anomalies to faults

In addition to these points, the techniques were crucial for the author's own research during this PhD. The development of an approach that works towards autonomous detection of anomalies would not have been possible without the visual data mining techniques. Only by their application was it possible to analyse the results, understand and evaluate the classifier's outputs, and understand the available training and test sets as will be shown in Section 9.3.2.

Chapter 5

Anomaly detection as a classification problem

This chapter discusses anomaly detection using classification techniques that learn from training data. The topic machine learning is surveyed, the fundamentals of classification theory are given, and common classifiers are introduced. Finally the shortcomings of two-class classification for the problem discussed in this Thesis are identified, and consequently one-class classification is introduced.

As concluded in the previous chapter, a user-driven approach is error-prone and does not scale well for large data bases of recordings. The underlying idea to reach **Aim 1** of this Thesis (Section 1.4), pointing the expert to potential errors in test drive recordings, is not to pre-configure search criteria, but rather extract knowledge from available recordings from test drives and then to autonomously detect anomalies in unseen data sets. This demands an approach capable of learning from sample data. Therefore in this chapter the field of machine learning is introduced and anomaly detection is discussed as a classification problem.

5.1 Machine learning – learning from sample data

Following the definition of (Mitchell, 1997), machine learning is concerned with the construction of computer programs that automatically improve with experience – i.e. computer programs having the ability to learn. The tasks of machine learning are manifold, e.g. classification, clustering, or prediction. This chapter focusses on the task of classification.

Definition 5.1 Learning system*: A system is said to have the ability to learn if for a given task T, the performance P improves with experience E (Mitchell, 1997).*

Applied to the field of anomaly detection, the task T corresponds to the detection of anomalies. The performance measure P expresses a system's ability to classify unseen data instances correctly. The term experience E given in the definition corresponds to the ability for generalisation from a given training data set. Best accuracy on unseen data is reached, when the underlying process of the data is learnt rather than the specifics of the training data set, a problem referred to as overfitting. When a highly flexible decision boundary is learnt that tightly encloses each individual instance, the decision boundary is likely to be overfitted to the training set.

The general steps in a machine learning system are illustrated in Figure 5.1. The first step is the data acquisition step, which comprises obtaining and selection of data sets. Optionally the input data can be pre-processed, for example by resampling or removal of invalid data. Following that, a further optional step is the transformation of the data set to an alternative representation that, for example, allows for better distinction of classes or for more efficient processing. Examples are the mapping of the floating point values of a time series to a limited number of digits or symbols, e.g. using SAX (Keogh et al., 2006), or the transformation to frequency domain using Fourier or wavelet transformation. Subsequently, a vital step is the extraction of features, a machine learning algorithm can work on. As a final step, the features relevant for the given machine learning task are selected.

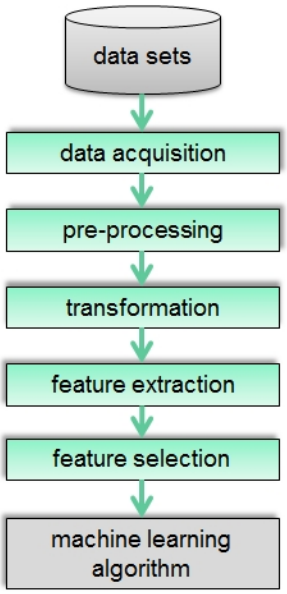


Figure 5.1: General steps in a machine learning system ranging from the acquisition of data to the application of a machine learning algorithm.

The described steps are conducted in the so-called training period, where knowledge is extracted from a training data set. The goal is to extract knowledge in a way to be able to classify unseen data. Subsequently a test period is conducted using a test data set containing instances with known class labels and the classification results are evaluated. The training and test period may be repeated various times in order to optimise parameters or the constitution of the training data set. Based on the extracted knowledge, unseen data is classified as normal or abnormal in the performance period. The training data set in feature space is denoted by $\mathcal{A}$, the test data set by $\mathcal{B}$, and the unseen data set by $\mathcal{C}$.

Machine learning algorithms are typically not applied on the raw input data, but rather on extracted features. Features are an abstraction of the original input data, characterising the data.

A data set in the original input space, i.e. the raw data set, is denoted by $\mathbb{I}$ and a data set in feature space is referred to as $\mathcal{F}$. The mapping function from original input space to feature space, i.e. feature extraction, is denoted by ϕ. The mapping from input to feature space is hence formulated as

$$\mathcal{F} = \phi(\mathbb{I}) \tag{5.1}$$

An instance in the original input space is denoted by $\mathbb{I}_v$ and in feature space by $\mathcal{F}_v$. $\mathcal{F}_v$ is referred to as a feature vector, which is l-dimensional, i.e. it contains l features f.

An example of an instance in input space $\mathbb{I}_v$ would be a univariate time series represented by its raw values. Mapping to feature space could be done by extracting statistical features (Mitsa, 2010), e.g. mean value and standard deviation. The representation of $\mathbb{I}_v$ in feature space could then be the feature vector $\mathcal{F}_v = [\mu, \sigma] = [3.42, 2.25]$. In that case, the mapping function $\mathcal{F}_v = \phi(\mathbb{I}_v)$ corresponds to the calculation of μ and σ.

Features extractable from time series data can be categorised into statistical and structural features (Olszewski, 2001). Statistical features are based upon quantitative properties of the data, e.g. mean value, standard deviation, minimum/maximum value, median or binned frequency counts (histograms). Structural features on the other hand describe the organisation of subpatterns (e.g. shapes) and their interrelationships in the data. The ordering of the data is taken into consideration.

Adding more distinctive features contained inside the data during a machine learning task can improve the classification accuracy, until a certain point is reached, where the accuracy actually decreases. This is due to the phenomenon referred to as curse of dimensionality which was found by Richard Bellmann (Bishop, 1995). As a rule of thumb, adding dimensions requires the number of training samples to grow exponentially, to avoid performance reduction. The number of training samples is often limited, though.

5.2 Two-class classification

Classification is the task of assigning an instance $\mathcal{C}_v$ to one of k classes ω_c. The number of classes and their labels are known beforehand.

The problem of anomaly detection can be viewed as a two-class classification problem (Chandola et al., 2009). The task is to assign an unclassified instance from $\mathcal{C}$ to either the normal class ω_n or the abnormal class ω_a based on a set of features f. Hence, this chapter discusses two-class classification with the target class ω_n and the outlier class ω_a.

The task of classification can also be described as learning a function that maps input variables to a pre-defined set of output variables, i.e. the class labels. An example would be the classification of a sensor signal as either normal or abnormal based on the features mean value, standard deviation, minimum and maximum values.

The accuracy of classification results is expressed with a confusion matrix (Fawcett, 2004) based on classification results obtained from classifying a test data set with known class labels as shown in Table 5.1. The confusion matrix consists of the true negatives (TN), which in accordance with (Tax, 2001) is the number of abnormal instances correctly classified as abnormal, the false positives (FP), the false negatives (FN), and the true positives (TP).

Confusion matrix		
	Classification result	
Class label	Normal	Anomaly
Normal	TP	FN
Anomaly	FP	TN

Table 5.1: Confusion matrix showing classification results with: TN = true negatives, i.e. an anomaly classified as abnormal, FP = false positives, FN = false negatives and TP = true positives.

Based on the confusion matrix, a variety of measures can be deduced. In this Thesis, in addition to the results in the confusion matrix, the following measures from (Fawcett, 2004) are utilised:

- *True positive rate*: The true positive rate (TPR) indicates whether a positive instance is classified as positive. A true positive rate of 100% means that all positive instances are classified as positive. It does not consider the fraction of instances falsely classified as positive, though. It is calculated by

$$TPR = \frac{TP}{TP + FN} \tag{5.2}$$

- *True negative rate*: The true negative rate (TNR) gives the percentage of detected anomalies. It is given by

$$TNR = \frac{TN}{TN + FP} \tag{5.3}$$

- *Precision*: The precision expresses the percentage of true anomalies in the result set of all instances classified as anomaly. If all instances classified as abnormal are anomalies, the precision is 100%. It makes no statement about what fraction of anomalies were falsely classified as normal, though.

$$precision = \frac{TN}{TN + FN} \tag{5.4}$$

Anomaly detection techniques based on classification approaches can be categorised based on the properties of the training data set. The training data set can either contain labelled instances from both classes – normal (ω_n) and abnormal (ω_a) – or only from one of the classes, typically from the normal class. Additionally a training data set may contain unlabelled instances only. This leads to the categorisation as given in (Hodge and Austin, 2004; Chandola et al., 2009):

- *supervised anomaly detection*: The training data set contains labelled instances from the normal and abnormal class. Unseen data instances are classified as normal or abnormal.

- *semi-supervised anomaly detection*: The training data set contains only normal instances. The goal is to learn the normal behaviour. Data instances deviating from the learnt normal behaviour are classified as anomalies.

- *unsupervised anomaly detection*: The training data set contains unlabelled instances only. The underlying assumption is that normal instances are much more frequent than anomalies. The frequent instances are classified as normal.

5.2.1 Fundamentals of classification

Statistical classification based on the Bayes theorem forms the fundamentals of classification theory. The goal of Bayesian learning is to minimise the probability of misclassification (Bishop, 1995). It is a probabilistic approach to machine learning based on the assumption that the data obeys probability distributions. Either the probability distributions are known or they are estimated (Mitchell, 1997). For a given, unclassified instance $\mathcal{C}_v$, classification is conducted by determining the most probable class ω_c using the Bayes theorem:

$$p(\omega_c|f) = \frac{p(f|\omega_c)P(\omega_c)}{P(f)} \tag{5.5}$$

- $p(\omega_c|f)$: the posterior probability, i.e. the probability that when observing a feature value of f the instance is of class ω_c

- $p(f|\omega_c)$: the probability density function (*pdf*), i.e. the probability of observing the feature value f in instances of the class ω_c, also referred to as the likelihood (Duda et al., 2001)

- $P(\omega_c)$: prior probability that an instance of class ω_c will be observed in the data set, also termed as the class-conditional probability (Duda et al., 2001)

- $P(f)$: prior probability that a given feature value of f will be observed, also referred to as the evidence (Duda et al., 2001)

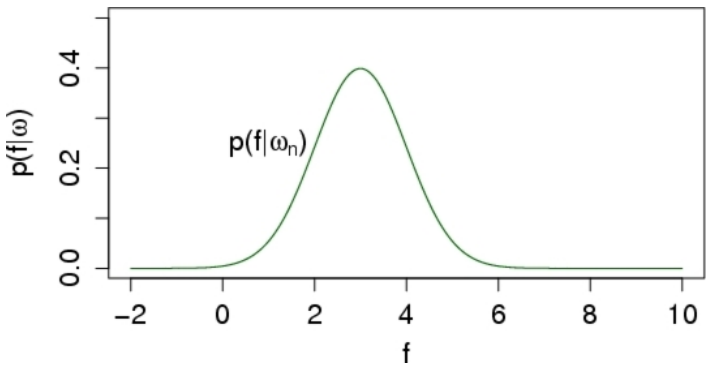


Figure 5.2: Probability density function $p(f|\omega_n)$ for one class and one feature.

Contemplating one class ω_n with one feature f and assuming a normal distribution (Gaussian distribution) leads to a probability density function (*pdf*) as shown in Figure 5.2.

The probability density function $p(f|\omega_n)$ shows how the values of the feature f are distributed for class ω_n, or formally stated: Given an instance is of class ω_n, what is the probability to observe a specific feature value f. The *pdf* $p(f|\omega_n)$ sums up to 1 as given by $\int_{-\infty}^{+\infty} p(f|\omega_n)\,\mathrm{d}f = 1$.

5.2.1.1 Probability density functions of two classes

An example of probability density functions of a normal class ω_n and an anomaly class ω_a are illustrated in Figure 5.3. Instances can be classified by contemplating the feature value f and assigning the class ω_n for feature values $< f_0$ and the class ω_a for feature values $> f_0$.

The case $f = f_0$ can be handled either by arbitrary class assignment, by favouring one class, or by rejection of a classification result. Yet, assuming a real-valued feature f which of the suggested solutions is chosen is not very relevant for the classification accuracy in practical considerations.

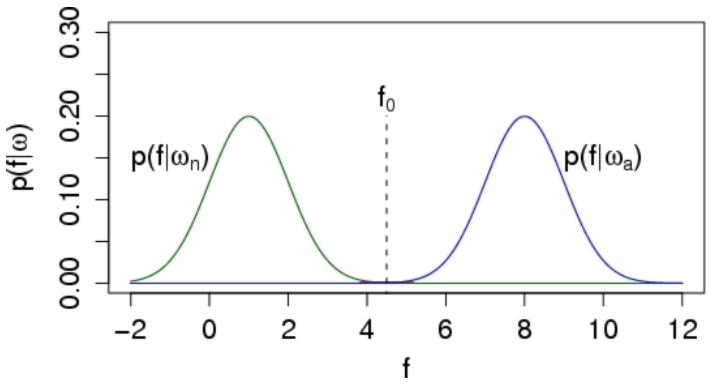


Figure 5.3: Probability density functions of two classes ω_n and ω_a with no apparent class overlap

5.2.1.2 Probability density functions of two overlapping classes

In Figure 5.4 the probability density functions of the two classes ω_n and ω_a are considered. The decision boundary is the feature value f_0, where the two *pdf*s intersect. For feature values close to f_0, the instance could be of both classes, because there is a region where the *pdf*s overlap.

The Bayesian classifier discriminates the two classes based on the feature value f by classifying the instance w.r.t. the maximum a posterior probability (MAP), thereby minimising the probability of misclassification:

$$\frac{p(\omega_n|f)}{p(\omega_a|f)} \quad \begin{cases} > 1 & \omega_n \\ < 1 & \omega_a \end{cases} \tag{5.6}$$

5.2.1.3 Probability density functions of two massively overlapping classes

In Figure 5.5 the *pdf*s of two classes with massive class overlapping are shown. Obviously the classification accuracy significantly decreases compared to the previous examples.

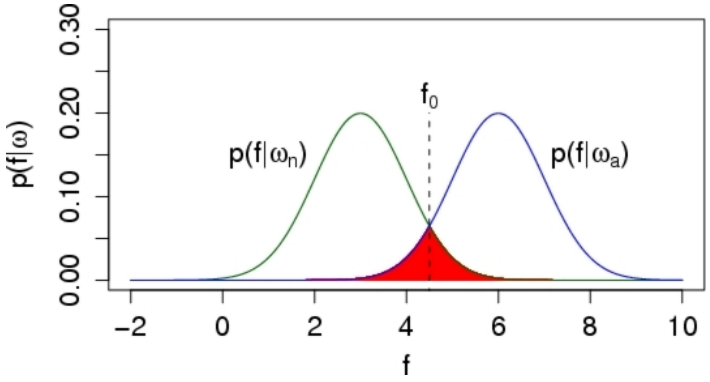


Figure 5.4: Overlapping probability density functions of two classes ω_n and ω_a

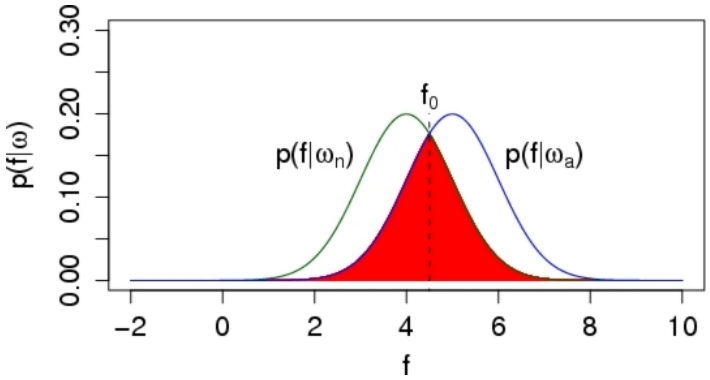


Figure 5.5: Massively overlapping probability density functions of two classes ω_n and ω_a

.

The classification accuracy may be unsatisfactory, so the question arises, what the options are to improve classification accuracy for the two given classes. A vital fact is that it is not a matter of finding a better classifier, as the class overlap is independent of the classifier. The problem is rather that the selected feature does not sufficiently discriminate the two classes ω_n and ω_a, or simply stated, the instances cannot be classified well based on the one observed feature.

The solution is to add additional features to the classification problem. Only in trivial classification problems, will one in fact be able to find a single feature that separates the classes sufficiently well.

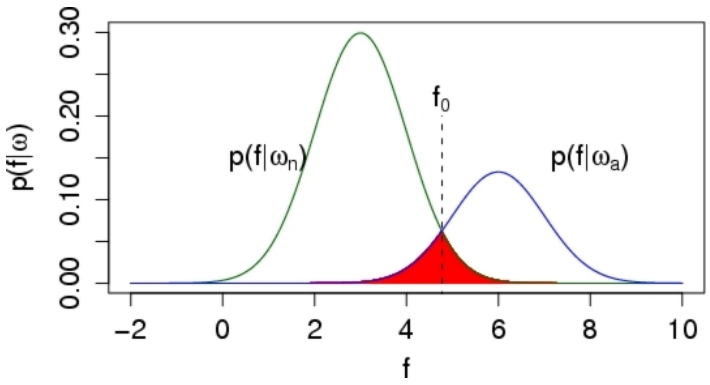


Figure 5.6: Probability density functions of two non-equiprobable classes ω_n and ω_a

5.2.1.4 Probability density functions of two non-equiprobable classes

Finally, classes with unequal probabilities are considered. The two classes in the previous example shown in Figure 5.4 are equiprobable, which means that the probability of observing an instance from class ω_n and ω_a is equal, i.e. $p(\omega_n) = p(\omega_a) = 0.5$. This is referred to as a balanced classification problem.

In many classification problems this is not the case. This is particularly true for the task of anomaly detection. The occurrence of anomalies (ω_a) is typically much less probable than the occurrence of normal behaviour (ω_a), i.e. $p(\omega_n) \gg p(\omega_a)$. Anomaly detection corresponds to an imbalanced two-class classification problem. Figure 5.6 shows the same *pdf*s as given in Figure 5.4, but with unequal prior probabilities $\frac{p(\omega_n)}{p(\omega_a)} = \frac{3}{2}$. As shown the prior probabilities influence the decision boundary f_0.

5.2.1.5 The Bayesian error

The region where instances are misclassified by the Bayesian classifier using the decision boundary f_0 as shown in Figures 5.4 - 5.6 is referred to as the Bayesian error e_{Bayes}.

Misclassification takes place where the two probability density functions overlap. The probability of misclassification, i.e. the Bayesian error e_{Bayes}, corresponds to the in-

tersection area of the two probability density functions. Hence the Bayesian error can be formulated as the following integral (Theodoridis and Koutroumbas, 2009):

$$e_{Bayes} = \frac{1}{2}\int_{-\infty}^{f_0} p(f|\omega_a)\,\mathrm{d}f \quad + \quad \frac{1}{2}\int_{f_0}^{+\infty} p(f|\omega_n)\,\mathrm{d}f \tag{5.7}$$

Generally, the error rate of any classifier cannot be below the Bayesian error e_{Bayes}. However, in an experiment, the measured error rate may be below the Bayesian error for a small number of instances by randomly guessing correctly, but the error rate will converge to the Bayesian error e_{Bayes} for a larger number of classified instances.

The Bayesian error e_{Bayes} lower bounds the error rate for any classifier. It is important to note that the error is formulated independently of classifier properties, the error exclusively depends on the properties of the data set in feature space $\mathcal{F}$.

5.2.1.6 Discussion

In this section, the Bayesian error and classification based on the Bayes theorem was introduced. It was shown that a classifier assigning the maximum a posteriori probability according to the Bayes theorem is optimal in a sense that no classifier can reach a higher classification accuracy in the given feature space $\mathcal{F}$.

So the question arises, why one would use any other type of classifier. The answer is that some strong assumptions were implicitly made in the introduction of the Bayesian framework. The calculation of the Bayesian error e_{Bayes} according to eq. (5.7) as well

as classification based on the Bayes theorem given in eq. (5.5), postulates that the following properties are known:

1. the type of probability density functions
2. the statistical parameters of the probability distributions
3. the prior class probabilities

In the examples, a normal (Gaussian) distribution with known parameters μ and σ was assumed. In practical applications the true distribution is typically not known, because this would mean that the underlying process that generated the data is fully understood and modelled mathematically correct. If this is the case, one would not use a machine learning approach, but rather base the classification on the mathematical model.

Further, the prior class probabilities have to be known for all classes, i.e. the true probability that an instance is of class ω_c. Thinking of one class as the anomaly class, the class containing the potential errors, this would mean that the true probability of a potential error has to be known a priori. This is unlikely to be known in practice.

So, is classification based on the Bayes theorem just a theoretical framework proving to be useless in practical applications?

In fact, the Bayesian framework is fundamental for understanding classification. The notion of the Bayesian error e_{Bayes} is useful, since it forms a lower bound for the error rate that is classifier-independent. Also, there are situations where the required a priori knowledge is known: For the evaluation of classifiers, often artificial data sets are used, that were generated following some probability distribution. In that case, the error rates of the evaluated classifiers can be compared to the Bayesian error.

Also for practical applications, classification based on the Bayes theorem can be used. The required a priori knowledge, i.e. the probabilities, can be estimated from the data set. The prior probabilities are estimated based on the frequency of instances of the classes ω_n and ω_a. The probability density function is estimated by assuming some type

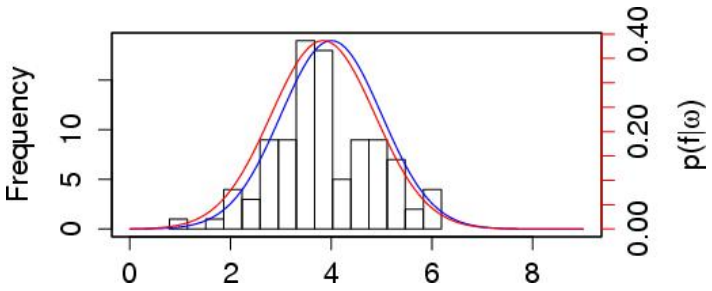


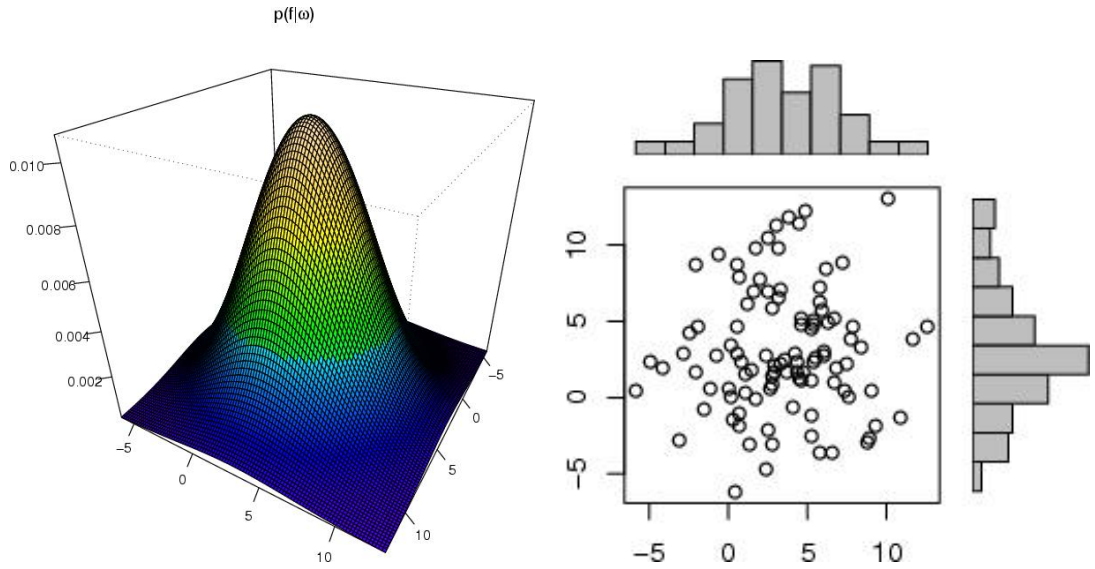
Figure 5.7: From a given probability density function (blue) a data set is generated. From the generated data set a histogram is calculated, and the probability density function is estimated (red).

of *pdf* or testing several candidate *pdf*s. However, estimating probability distributions becomes very challenging in high-dimensional space. Instances in high-dimensional space are sparsely distributed, hence a high number of instances are required.

Based on these estimations, classification can be conducted and the Bayesian error can be calculated. Since this error calculation is based on estimations, it is not equal to the true Bayesian error e_{Bayes}. The quality of the classification and the accuracy of the estimated Bayesian error depends on the quality of the estimations. If the estimations are poor, the classification results will be poor as well.

In an experiment, estimating probabilities of one class is done on a training data set $\mathcal{A}$ with 100 instances. A normally distributed 1-dimensional data set is generated with $\mu_{generated}$ and $\sigma_{generated}$. Now the values $\mu_{estimated}$ and $\sigma_{estimated}$ are calculated from the generated data set assuming a normal distribution. The true and the esitmated *pdf* are shown in Figure 5.7. The bigger the training data set, the better the estimation will be, but in general $\mu_{generated} \neq \mu_{estimated}$ and $\sigma_{generated} \neq \sigma_{estimated}$.

Estimating probabilities becomes harder for higher dimensional data sets, which can be illustrated even for 2 dimensions, by contemplating a generated 2-dimensional data set with two independent normal distributions, i.e. the covariances are 0. The underlying *pdf* is shown in Figure 5.8(a). The properties of the generated data set differ as can be seen in the 2-D histogram determined from the generated data as shown in Figure 5.8(b). One could correctly assume two normal distributions, but based on the histogram, other probability distributions could be estimated as well.



(a) Probability density function of a two-dimensional data set with independent normal distributions.

(b) Histograms of a generated two-dimensional data set following an underlying probability density function.

Figure 5.8: Estimation of the probability density function from a generated data set.

It should be noted that there are more advanced methods for estimating the *pdf*, instead of just assuming a normal distribution (Duda et al., 2001), e.g. using a mixture of Gaussians.

5.2.2 Linear classifiers for anomaly detection

Linear classifiers work by determining a linear function that separates the classes ω_n and ω_a in the training data set $\mathcal{A}$ and classify an unseen instance $\mathcal{C}_v$ by determining on which side of the decision function it is. Decision functions will be denoted by $d(\mathcal{F})$. Some linear classifiers, applicable to the detection of anomalies are discussed in this section.

Functioning of a basic linear classifier (shown in Figure 5.9):

1. determination of a linear decision function $d(\mathcal{F})$ from the training data set $\mathcal{A}$
2. classification of an unseen instance $\mathcal{C}_v$ is done by determining on which side of the decision function $d(\mathcal{F})$ the instance resides

A linear decision function in l-dimensional feature space has the dimension $l-1$. So, the linear decision functions correspond to a straight line in a 2-dimensional feature space, a plane in 3-dimensional space and a hyperplane in higher-dimensional space. A linear decision function in 2-dimensional space (Figure 5.9) is described by:

$$d(\mathcal{F}) = w_1 f_1 + w_2 f_2 + w_0 \tag{5.8}$$

where w_1 and w_2 are the weights[1], f_1 and f_2 are the dimensions in feature space and w_0 is the bias corresponding to the distance of the decision function from the origin. In literature, the bias is also denoted by b.

Generalising eq. (5.8) to l dimensions is achieved by combining $w_1..w_l$ to the transposed vector of weights denoted by w^T and $f_1...f_l$ to a vector of features $\mathcal{F}$:

$$d(\mathcal{F}) = w^T \mathcal{F} + w_0 \tag{5.9}$$

[1]Please note that $w \neq \omega$. w denotes the weights and ω denotes the classes, in accordance with the common notation in literature on classification (Duda et al., 2001; Theodoridis and Koutroumbas, 2009; de Sa, 2001).

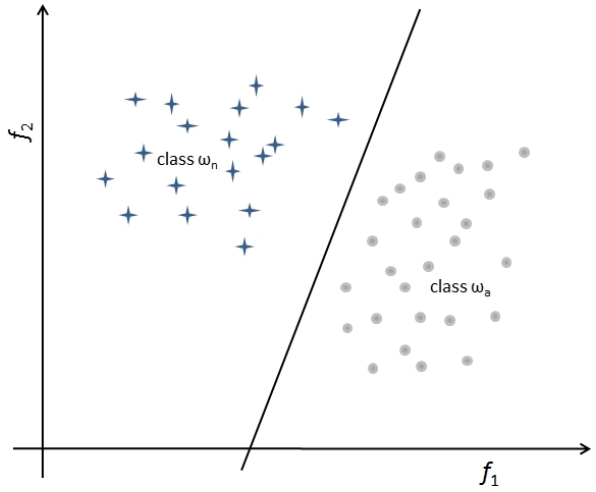


Figure 5.9: Linear classifier separating the normal class ω_n from the abnormal class ω_a

Having determined a decision function $d(\mathcal{F})$, classification takes place by calculating on which side of $d(\mathcal{F})$ an instance $\mathcal{C}_v$ is by solving $d(\mathcal{C}_v)$.

In the rest of this section the common linear classifiers, nearest mean classifier, Fisher classifier and linear support vector machines, are introduced.

5.2.2.1 Nearest mean classifier

The nearest mean classifier works by determining the classes' mean value from the training data set $\mathcal{A}$ and classifying an unseen instance $\mathcal{C}_v$ to the class with the nearest mean value. The linear decision function $d_{nmc}(\mathcal{F})$ is determined by connecting the two class means μ_{ω_n} and μ_{ω_a} and finding the decision function that orthogonally intersects the connecting line at $\frac{|\mu_{\omega_n} - \mu_{\omega_a}|}{2}$.

Functioning of nearest mean classifier:

1. determine mean value of classes ω_n and ω_a from the training data set $\mathcal{A}$

2. for an unclassified instance $\mathcal{C}_v$, find the class ω_c that has the nearest mean value and assign $\mathcal{C}_v$ to that class

5.2.2.2 Fisher classifier

The Fisher classifier was introduced in (Fisher, 1936), an early and fundamental work in classifying data based on a training data set. The classifier works by finding a linear decision function that maximises the fraction of the variance between classes over the variance within the classes in the training data set $\mathcal{A}$. If the variances are equal, the Fisher classifier corresponds to the nearest mean classifiers. The decision function is given by (Duda et al., 2001):

$$d_{Fisher}(\mathcal{F}) = max(\frac{|\mu_{\omega_n} - \mu_{\omega_a}|^2}{\sigma^2_{\omega_n} + \sigma^2_{\omega_a}}) \tag{5.10}$$

Functioning of Fisher classifier:

1. determine mean value and standard deviation of classes ω_n and ω_a from the training data set $\mathcal{A}$

2. determine the linear decision function utilising eq. (5.10)

3. classify an unseen instance $\mathcal{C}_v$ using $d_{Fisher}(\mathcal{F})$

5.2.2.3 Support vector machine as a linear classifier

A support vector machine (SVM) can be used for two-class classification and thereby for anomaly detection. Support vector machines (Abe, 2010; Han and Kamber, 2006; Theodoridis and Koutroumbas, 2009) have shown to yield good results for classification problems. In a classification problem of two classes, they determine a hyperplane separating the classes.

An SVM is a so-called maximum margin classifier. The separating hyperplane is determined from the training data set $\mathcal{A}$, by demanding a class separation with maximum margin. No assumptions about probability distributions are made. Classification is exclusively done based on the instances at the boundaries of the classes.

The way of finding the decision function is introduced by using the 2-dimensional data set shown in Figure 5.10. The first step is finding two parallel lines $g_1(\mathcal{F})$ and $g_2(\mathcal{F})$ intersecting one or more of the instances at the boundaries of the two classes ω_n and ω_a. The number of possible line pairs is infinite, so a further criterion is needed in order to find a unique decision function. SVMs find the one decision function that maximises the margin between the classes. This is done by finding those two parallel lines with maximum distance to each other. The optimal decision function $d_{SVM}(\mathcal{F})$ is now the line that is in the middle of the two parallel lines $g_1(\mathcal{F})$ and $g_2(\mathcal{F})$. In higher-dimensional feature spaces the decision function is a plane or hyperplane instead of a line.

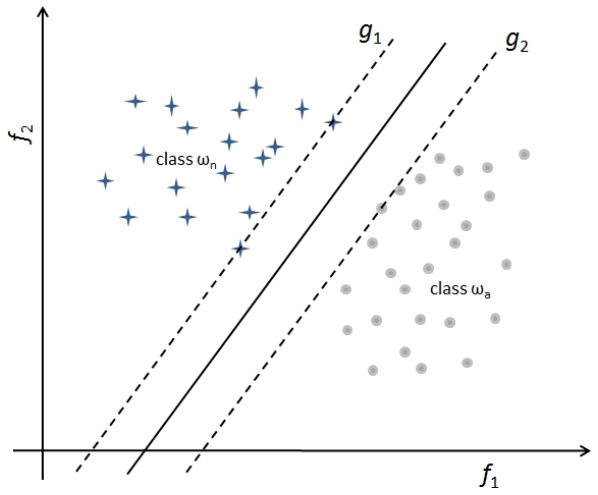


Figure 5.10: Linear decision function determined by a hard-margin support vector machine. $g_1(\mathcal{F})$ and $g_2(\mathcal{F})$ are illustrated with dashed lines.

Formulating $g_1(\mathcal{F})$ and $g_2(\mathcal{F})$, w and w_0 are scaled so that their distance to the separating hyperplane is equal to ± 1:

$$\begin{aligned} g_1(\mathcal{F}) &= w^T\mathcal{F} + w_0 \geq +1 \quad \text{for all instances in } \omega_n \\ g_2(\mathcal{F}) &= w^T\mathcal{F} + w_0 \leq -1 \quad \text{for all instances in } \omega_a \end{aligned} \tag{5.11}$$

The decision function $d_{SVM}(\mathcal{F})$ is that hyperplane, that is parallel to and has equal distances to $g_1(\mathcal{F})$ and $g_2(\mathcal{F})$. The linear decision function can be described by:

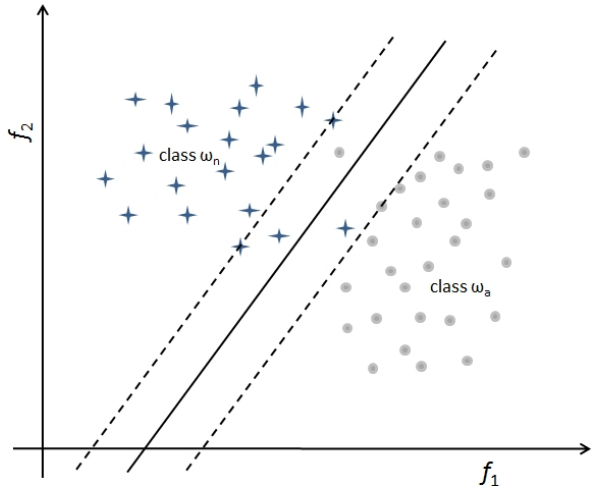


Figure 5.11: Soft-margin support vector machine allowing some instances in the training data set to be misclassified

$$d_{SVM}(\mathcal{F}) = w^T \mathcal{F} + w_0 \tag{5.12}$$

A major advantage of SVMs is that finding the optimal decision function $d_{SVM}(\mathcal{F})$ leads to a convex optimisation problem (Abe, 2010). As opposed to for example the parameter tuning in neural networks, in finding the separating hyperplane there are no local minima: there is only one hyperplane separating the training data set with a maximum margin as shown in Figure 5.10.

If the classes in $\mathcal{A}$ are not 100% separable, the hard-margin support vector machine will not find a solution, as no linear decision function can be found. This means that one instance in $\mathcal{A}$, possibly an outlier, can prevent a support vector machine to be applicable. This problem is solved by relaxing the constraint given by eq. (5.11) of separating *all* instances in the training data set correctly. This is achieved by introducing slack variables, allowing some instances to be outside the decision boundary. This type of support vector machine is referred to as a soft-margin SVM, shown in Figure 5.11.

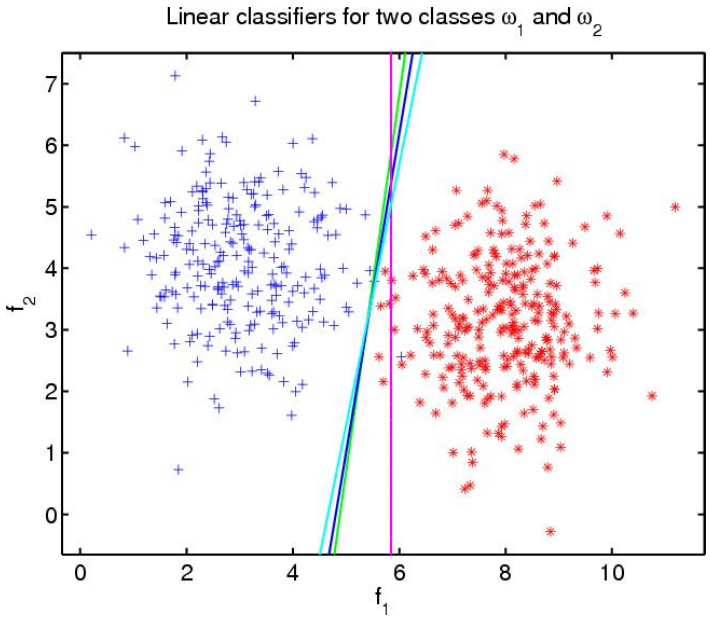


Figure 5.12: Example of linear classifiers: Fisher classifier (green), nearest mean classifier (blue), decision stump (magenta), and linear support vector machine (cyan) applied to separate the normal and abnormal class obeying Gaussian distributions (created with (PRTools, 2012)).

5.2.2.4 Classification examples of linear classifiers

The application of different linear classifiers and their differences are shown in Figure 5.12 by linearly separating the two classes ω_n and ω_a in an artificially generated 2-dimensional data set $\mathcal{A}$. The instances of the two classes obey normal distributions with standard deviations of $\sigma_{\omega_n} = \sigma_{\omega_a} = 1$ and mean values of $\mu_{\omega_n} = (3, 4)$, $\mu_{\omega_a} = (8, 3)$.

As can be seen in Figure 5.12, different classifiers have different decision functions on the same data set $\mathcal{A}$. In practical applications the challenge lies in finding a classifier, that best matches the properties of a data set. However, the task is not to optimally classify the training data set $\mathcal{A}$ but rather to optimally classify unseen instances $\mathcal{C}_v$.

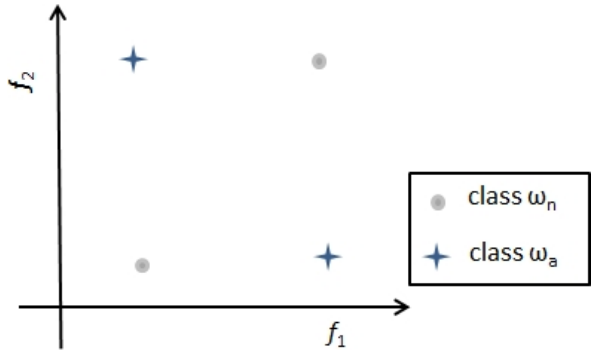


Figure 5.13: The XOR problem shows two classes in two-dimensional feature space that are not linearly separable.

5.2.3 Non-linear classifiers for anomaly detection

The benefit of linear classifiers is their simplicity and their compact representation of the decision boundary. Classification corresponds to determining on which side of the linear decision function an unseen instance is, allowing for very fast classification.

However, classes are often not linearly separable in the given feature space $\mathcal{F}$. An example is the so-called XOR problem (Bishop, 1995), shown in Figure 5.13. The four instances, two of each class ω_n and ω_a, are not linearly separable.

In general there are two approaches to separate classes that are not linearly separable:

1. using a classifier capable of learning a non-linear decision function

2. mapping the feature space to a higher-dimensional feature space, where the classes can be linearly separated

An example of a classifier able to learn a non-linear decision function would be a polynomial classifier of degree > 1 like a quadratic decision function, or a nearest neighbour classifier which can build arbitrary non-linear decision functions. Other types of classifiers map the feature space to a higher dimension prior to classification

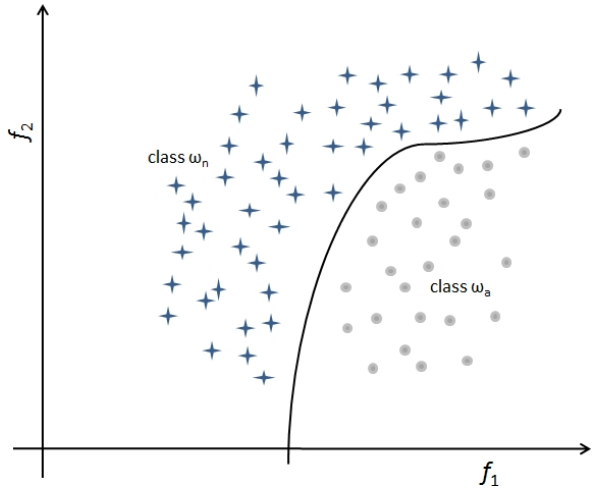


Figure 5.14: Example of a non-linear decision function separating classes ω_n an ω_a.

by introducing additional features, derived from the existing features. An example of such a classifier is a support vector machine using so-called kernels (Abe, 2010).

An example of a non-linearly separable data set separated by a non-linear decision function is shown in Figure 5.14. Common non-linear classifiers are introduced in the rest of this section.

5.2.3.1 Decision trees

The basic variant of a decision tree (Mitchell, 1997) tests each feature f at one node. Learning corresponds to building a tree with the challenge of determining what to test at which tree node, so-called decision tree induction (Han and Kamber, 2006). Two basic algorithms to learn a decision tree from a training data set are ID3 and C4.5 (Mitchell, 1997).

Anomaly detection can then be done by traversing the tree starting at the root node. The reached leaf node will then output either ω_n or ω_a.

5.2.3.2 k-nearest neighbour classifier

Due to its simplicity and its capability to learn arbitrarily formed decision boundaries, k-nearest neighbours (k-NN)(Mitchell, 1997; Theodoridis and Koutroumbas, 2009) is a well-known classifier. It works by storing all instances from the training data set $\mathcal{A}$ and classifying an unseen instance $\mathcal{C}_v$ with respect to $\mathcal{A}$. Classification is done by finding the nearest neighbours of $\mathcal{C}_v$ by calculating distances between $\mathcal{C}_v$ and all instances in $\mathcal{A}$. The test instance is classified according to the most frequent class in the found k neighbours.

No explicit decision function is calculated in the training period. Classification is exclusively done by determining the nearest neighbours for each $\mathcal{C}_v$. For this reason, nearest neighbour classifiers are also referred to as lazy learners (Han and Kamber, 2006). Widely used nearest neighbour classifiers are 1-NN, k-NN, or weighted k-NN (Mitchell, 1997).

Functioning of the basic k-nearest neighbour classifier as illustrated in Figure 5.15:

1. store all instances from training data set $\mathcal{A}$
2. select or determine the parameter k
3. for an unclassified instance, find the k nearest neighbours in $\mathcal{A}$ by calculating distances between the unclassified instance $\mathcal{C}_v$ and the instances in $\mathcal{A}$, as illustrated in Figure 5.15
4. classify $\mathcal{C}_v$ based on the majority class ω_c in the k nearest neighbours

5.2.3.3 Artificial neural networks

This section discusses artificial neural networks (ANNs) (Bishop, 1995). The basic variant is a multi-layer perceptron with backpropagation. The idea of neural networks

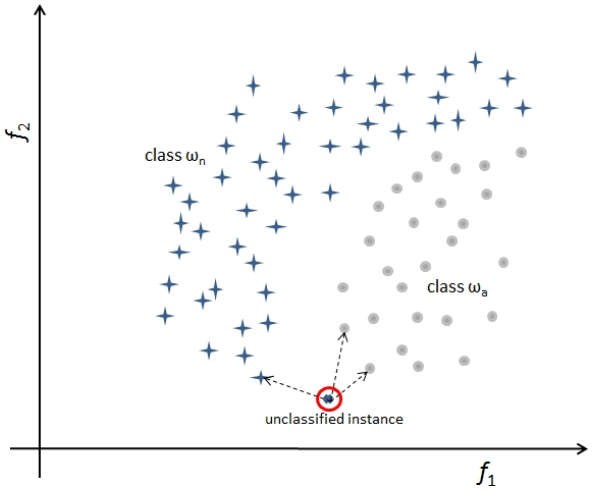


Figure 5.15: k-nearest neighbour classifier with $k = 3$: classifying an unseen instance by finding the 3 nearest neighbours. The test instance will be classified as ω_a, by a 2:1 vote.

is to mimic the way human beings think, by building a structure similar to the human brain, which consists of neurons interconnected by synapses.

A multi-layer perceptron neural network is a layered network of interconnected nodes called perceptrons. Each node has multiple inputs and a single output, where the nodes' outputs of layer N act as the inputs of the nodes of layer $N+1$. A node's inputs are aggregated using a weighted sum and are transformed to a single output using a so-called activation function. Common activation functions are the *sign* function to have a binary output or the *sigmoid* to have a real-valued output.

The network's topology is pre-configured, the parameters are the number of layers and the number of nodes per layer. Each input node processes one feature f, hence the number of input nodes is given by the dimension of the feature space $\mathcal{F}$. In classification tasks, the number of output nodes corresponds to the number of class labels. In anomaly detection, separating the normal class ω_n from the abnormal class ω_a can be achieved by an output layer with two nodes. The layers between input and output layer are referred to as hidden layers.

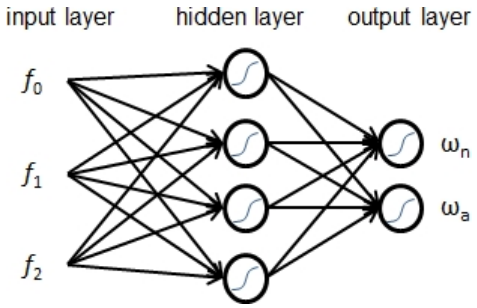


Figure 5.16: Exemplary topology of an artificial neural network with three layers, capable of classifying 3-dimensional feature vectors as either normal or abnormal

A single perceptron on its own can be used as a linear classifier. Neural networks with three layers are capable of learning arbitrary non-linear mappings. Figure 5.16 shows the layout of a neural network with three layers: three nodes in the input layer, one hidden layer and two output nodes [2].

The neural network is trained by iteratively classifying a fraction of the instances in the training data set $\mathcal{A}$ and determining the error e which is the difference between the network's output and the desired class labels. For a given data set and a given network topology, the error depends on the weights. Hence, learning the decision function is a matter of adjusting the network's weights w.

Using the backprogation technique to optimise the weights, the error is propagated to the network nodes starting at the output layer. The weights for the next run are determined using gradient descent in the direction with the highest decrease of the error:

$$w_{run} = w_{run-1} - \eta \frac{\partial e}{\partial w} \tag{5.13}$$

where w_{run} is the vector of weights for the current run, w_{run-1} the weights of the previous run, η is the learning rate, and $\frac{\partial e}{\partial w}$ are the partial derivatives of the error.

[2]The way the number of layers is specified is not consistent throughout literature. In this Thesis the input nodes are accounted as a layer, in accordance with (Duda et al., 2001). (Bishop, 1995) refers to the same network as a network with "two layers of adjustable weights".

Summarising, the functioning of neural networks using backpropagation is as follows:

1. pre-define the network topology and initialise the vector of weights w
2. classify a fraction of the instances from the training data set $\mathcal{A}$
3. determine the error rate e of the current run
4. adjust the weights w using gradient descent, in order to reduce the error rate
5. repeat the process to optimise the error rate until the optimum is found
6. classify an unseen instance $\mathcal{C}_v$, based on the pre-defined topology and the optimised weights w, by passing the instance to the input nodes

5.2.3.4 The support vector machine as a non-linear classifier

Support vector machines, that were introduced in Section 5.2.2.3 as linear classifiers, can be enhanced to function as non-linear classifiers. This is achieved by mapping the input feature space to a higher-dimensional feature space, where the data can be linearly separated by a hyperplane. Finding the decision function is then done as described in Section 5.2.2.3.

The functioning of such a mapping is explained with an example. Figure 5.17 shows a contrived XOR data set with two classes that look easily separable. However, it is impossible to linearly separate the two classes in the given feature space. For that reason a support vector machine, as described in Section 5.2.2.3, is not capable of learning a good decision function for this data set.

The data set can be mapped to a higher-dimensional space Z with a mapping function $\phi()$. In (Ben-Hur and Weston, 2010; Theodoridis and Koutroumbas, 2009) the following mapping was used as an example:

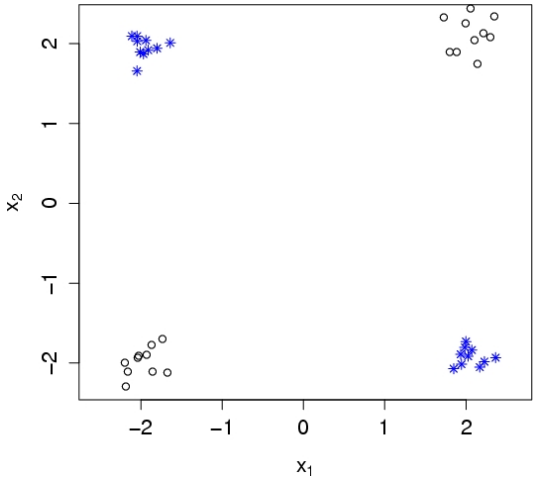


Figure 5.17: Contrived XOR data set with two classes (black points and blue stars), that are linearly inseparable.

$$\begin{aligned} z_1 &:= x_1^2 \\ z_2 &:= \sqrt{2} x_1 x_2 \\ z_3 &:= x_2^2 \end{aligned} \tag{5.14}$$

Using this mapping, the two-dimensional data set is mapped to a three-dimensional feature space as shown in Figure 5.18. It can be seen that the classes can now be linearly separated by a plane in the transformed feature space.

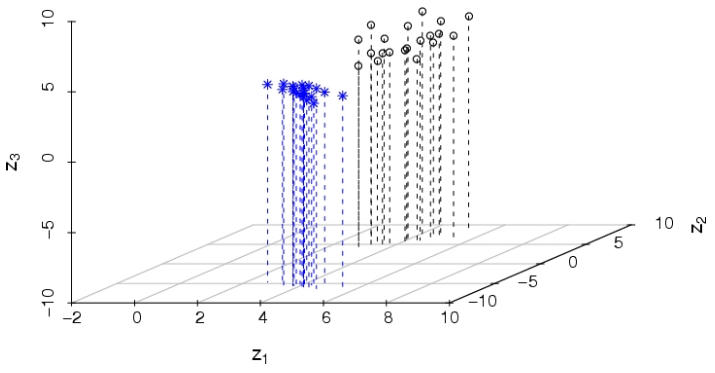


Figure 5.18: Two-dimensional XOR data set with two classes, that become linearly separable after mapping to three-dimensional feature space.

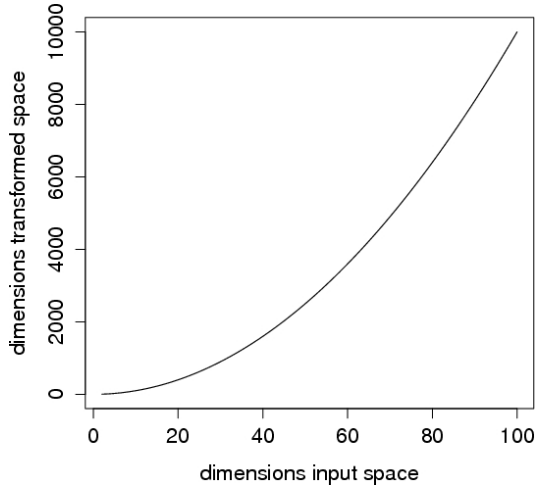


Figure 5.19: The mapping eq. (5.14) is quadratic in the number of dimensions in the transformed feature space.

Applying eq. (5.14) to an input space of 3 features, the transformed feature space has 6 dimensions $(x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_2x_3, \sqrt{2}x_1x_3)$. With the help of $\binom{n}{k}$, the number of dimensions $d_{transformed}$ w.r.t. the number of features d_{input} for this example mapping is given by

$$d_{transformed} = \binom{d_{input}}{2} + d_{input} \tag{5.15}$$

Although the mapping function was ideal for the given data set, it has some drawbacks.

- It is parameterless, so it is not adaptable to different data sets.
- From eq. (5.15) it can be seen that the mapping function does not scale well. It is quadratic in the number of dimensions in the transformed feature space as depicted in Figure 5.19.
- The mapping function $\phi()$ has to be known.

The so-called kernel trick (Theodoridis and Koutroumbas, 2009) is used to overcome these drawbacks. The weight vector w in the linear decision function given in eq. (5.12) is orthogonal to the hyperplane. As a first step, w is expressed as a linear combination of the training vectors, i.e. as $\sum \alpha_i x_i$:

$$d_{SVM}(\mathcal{F}) = \sum \alpha_i x_i^T \mathcal{F} + w_0 \tag{5.16}$$

Now the training instances and the vector of features are transformed by $\phi()$.

$$d_{SVM}(\mathcal{F}) = \sum \alpha_i \phi(x_i)^T \phi(\mathcal{F}) + w_0 \tag{5.17}$$

In eq. (5.17) $\phi(x_i)$ and $\phi(F)$ are present as inner products of the mapped vectors. The kernel trick is to replace the inner products by a kernel function. The mapping is then implicitly achieved by solving $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$.

A comprehensible introduction to support vector machines together with the kernel trick can be found in (Ben-Hur and Weston, 2010; Theodoridis and Koutroumbas, 2009). Advanced topics are covered in (Abe, 2010).

5.2.3.5 Classification examples of non-linear classifiers

In Figure 5.20 the non-linear decision functions determined by a quadratic classifier, a Parzen classifier and a k-NN classifier are shown. The decision functions learnt for the same training data set by a neural network, a decision tree and a support vector machine are shown in Figure 5.21.

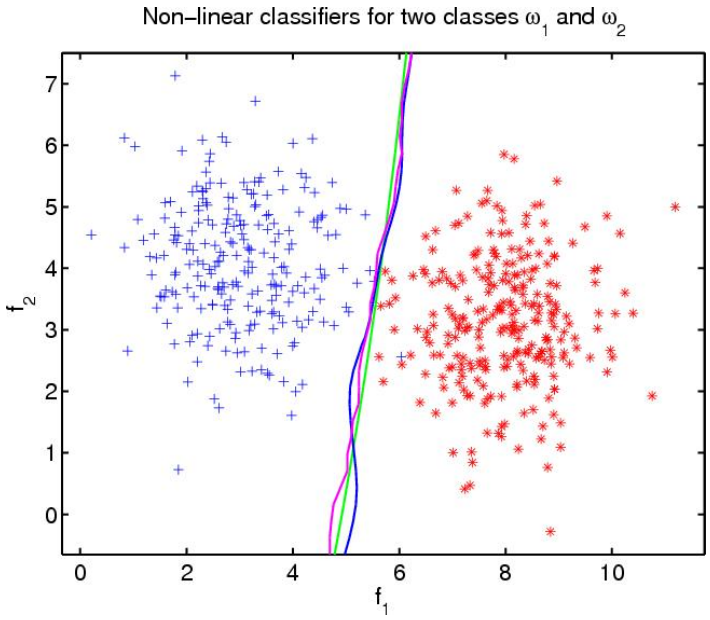


Figure 5.20: Example of non-linear classifiers: quadratic classifier (green), Parzen (blue), k-NN (magenta), created with (PRTools, 2012).

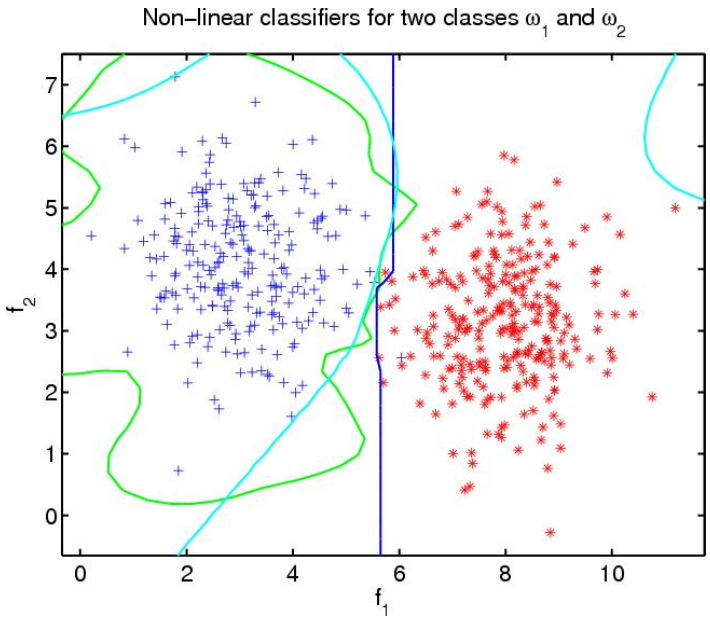


Figure 5.21: Example of non-linear classifiers: neural network (green), decision tree (blue), and non-linear support vector machine (cyan), created with (PRTools, 2012).

5.2.4 Discussion

The focus of this Thesis is the autonomous detection of anomalies in multivariate time series. It has been shown that machine learning approaches can be used to detect anomalies in multivariate time series (Chandola et al., 2009). According to the definition of anomalies, the goal is to detect unexpected behaviour. Using a machine learning approach this goal can be achieved by teaching the system normal and abnormal behaviour by means of normal and abnormal training data sets and have the system classify unseen data as either normal or abnormal. In the case of normal and abnormal corresponding to one class each, a two-class classification approach can be used.

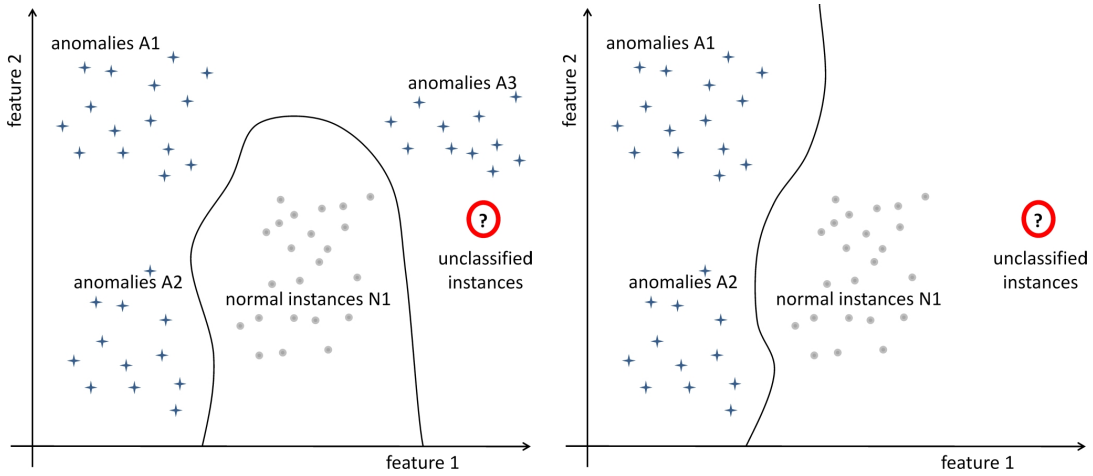
Two limitations of such a traditional classification approach for the problem discussed in this Thesis were identified:

1. *No abnormal data sets*: In many applications there are no abnormal data sets. On the other hand the amount of training data containing normal instances is not constrained, since the training data can be obtained by recording data from a system in normal operation mode.

2. *Non-representative abnormal data sets*: By using normal and abnormal training data sets, the classification is influenced by the choice of the abnormal data. Using a non-representative training data set of anomalies, an incorrect description of the abnormal class is learnt.

The consequence of the first limitation is obvious. A two-class classification cannot be conducted, if there is no training data from the abnormal class. The consequence of applying two class-classification in the case of a non-representative training data set, shall be illustrated in an example.

Say, the training is conducted on recordings from a vehicle subsystem, where three types of anomalies $A1$, $A2$ and $A3$ may occur. This is shown in a contrived two-dimensional feature space in Figure 5.22(a). The instance to be classified, represented by a question mark, will be classified as abnormal due to its closeness to $A3$. In real-



(a) Training set with representative instances from the abnormal class.

(b) Training set with missing instances from the abnormal class.

Figure 5.22: Fully representative and non-representative training set.

world applications, one cannot assume to have a training data set with representative anomalies, unless knowledge of all fault states of a system together with recordings of these fault states exists. If the training is conducted with a training set that only contains $A1$ and $A2$, the same instance will in this example be falsely classified as normal due to its closeness to the region of normal instances as shown in Figure 5.22(b).

Obtaining or creating abnormal training data is very intricate in many real-world applications. Normal data sets on the other hand can be obtained easily, e.g. by observing a system in all its normal operation modes over an arbitrarily long period of time.

To overcome the mentioned limitations of two class-classification approaches, the idea is to use a different approach. As opposed to learning normal and abnormal behaviour, only the normal behaviour shall be learnt, and deviations from this normal behaviour be classified as abnormal as shown in Figure 5.23. This classification technique is referred to as one-class classification, which will be discussed in the next section.

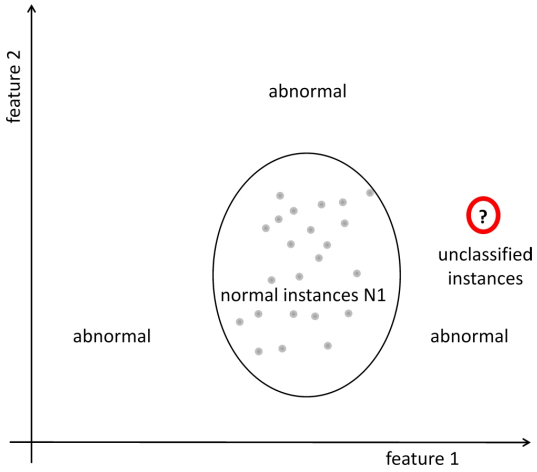


Figure 5.23: Feature space containing only normal instances

5.3 Anomaly detection as a one-class classification problem

In the previous section two-class classification was discussed. The training data set consists of instances from the normal and the abnormal class, allowing the boundary between the classes to be learnt using information from both of the classes. The boundary is thereby adjusted supported by instances from both classes.

For the detection of anomalies in test drives, there are no or non-representative abnormal instances available as training data. For that reason, the problem of anomaly detection is formulated as a one-class classification problem (Tax, 2001): from a training set $\mathcal{A}$ of labelled normal instances a classifier shall be learnt without contemplating instances labelled as abnormal. This is referred to as semi-supervised anomaly detection in (Chandola, 2009). The challenge in one-class classification is to learn the boundaries of the normal region, as the decision function will not be adjusted based on instances from two classes. Based on the learnt classifier, unseen instances are then classified as either normal ω_n or abnormal ω_a. So the question is:

Can unseen instances $\mathcal{C}_v$ be classified as either normal ω_n or abnormal ω_a by learning from a training data set $\mathcal{A}$ that exclusively contains normal data?

A major challenge is that optimising the trade-off between true and false positives is not possible using a test data set containing only normal instances, as in this case only true positives and false negatives are reported. Measuring true negatives postulates that anomalies are present in the test data set $\mathcal{B}$. Hence, in one-class classification problems, one inherent parameter is always present: a threshold. Classifying an unknown instance, its distance to the normal region is calculated. Based on this distance a threshold is required that separates between normal and abnormal. The threshold is a parameter, which can be

1. pre-defined by a user

2. fitted to yield a pre-defined number of anomalies as proposed in (Chandola, 2009), i.e. the expected fraction of anomalies has to be known

3. a data-driven threshold determined from the training data set $\mathcal{A}$, e.g. distances between the instances in the training data set

4. optimised with respect to an additional parameter

The fundamentals on statistical classification introduced in Section 5.2.1 are now applied to the problem of one-class classification.

The two overlapping classes from the previous section, that were represented by their probability density functions in Figure 5.4, shall be separated by only integrating knowledge about the normal class ω_n. The aim is to learn a closed boundary around the normal class, because potentially there could be anomalies on the left hand side of $p(f_p|\omega_n)$ as well. In Figure 5.24 the threshold (dashed lines) was arbitrarily set to $\pm 2\sigma$. It can be seen that this decision function is not as good w.r.t. the Bayes error, as it was the case when training with both classes.

In (Moya and Hush, 1996) the problem to exclusively learn normal behaviour and automatically classify unseen instances deviating from the learnt behaviour as either normal or abnormal was addressed and the term one-class classification was introduced. An architecture of three combined neural networks is proposed. One-class classification

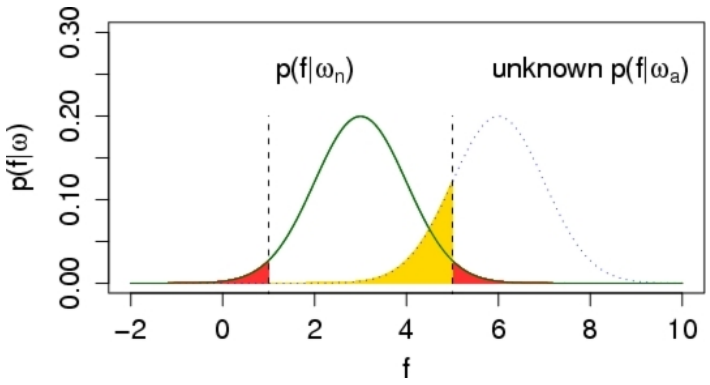


Figure 5.24: Decision function between ω_n and ω_a, with unknown ω_a. The yellow area corresponds to the false positives, the red areas to the false negatives.

using support vector machines was first addressed by Schoelkopf et al in (Schölkopf et al., 2001).

In (Chandola, 2009), Chandola contemplates the problem in the context of time series data and refers to the problem as semi-supervised anomaly detection. Based on an anomaly score, the test instances are ranked and the top p instances are reported as anomalies. Other publications refer to the problem as single class classification.

The techniques require a priori knowledge of the expected fraction of anomalies in the data in order to define a threshold, i.e. the decision boundary.

In (Tax, 2001), Tax discusses a way to automatically determine the threshold from the data. Based on support vector machines (SVM), an approach named support vector data description (SVDD) is proposed. A trade-off between the number of false positives and the volume of the normal region is proposed. On the one hand the region of normality is desired to capture as many normal training data sets as possible, while on the other hand keeping the region's volume at a minimum.

This Thesis aims to automatically classify unseen instances as either normal or abnormal based on a training data set of normal data only, without having to explicitly specify the fraction of expected anomalies a priori. The following key requirements for a classifier were identified:

1. The training period should be based on normal time series only.

2. The classifiers should be robust to noise in the training set, i.e. hidden anomalies in the training set.

3. Due to the complexity of the data, falsely reported anomalies (false negatives) are expected. Therefore a detected anomaly should have an anomaly score expressing the extent of the anomaly in order to be able to rank the reported anomalies.

4. An anomaly's time span should be reported.

5. During the operation time of the anomaly detection system, anomalies will be detected. Therefore the one-class classifier should be extensible by data from the abnormal class.

6. No machine learning expert should be required to adjust parameters.

5.4 Conclusion

This chapter discussed anomaly detection using classification techniques from the field of machine learning. Common linear and non-linear classifiers were introduced. Traditionally, classifiers are used for two-class or multi-class classification. It was argued that this is not suitable for the detection of anomalies in test drive recordings, since no representative training set of anomalies exists. Consequently, one-class classification was introduced, where a classifier is trained on a data set of normal instances.

The question is, which of the introduced classifiers satisfies the identified key requirements.

The linear classifiers can be ruled out due to their inability to separate arbitrary data. Hence, the non-linear classifiers are evaluated in the rest of this section. While all described classifiers are two- or multi-class classifiers, they can be adapted to work for the one-class case.

The benefit of decision trees is the human-readable representation. It is easy to interpret how the classifier came up with a result, by contemplating the traversed path. The decision function however is not very flexible, it is piecewise linear.

Standard k-nearest neighbour techniques require storing all learnt instances. While a huge training data set is beneficial in order to achieve good generalisation (Mitchell, 1997), storing all instances is particularly problematic for time series data due to the big amount of data per instance.

While k-NN is a simple classifier, arbitrarily-shaped boundaries can be learnt and the distribution of the data set does not have to be known. Following e.g. (Theodoridis and Koutroumbas, 2009), k-NN yields good classification results on training data sets with many instances. Another benefit is that theoretically a worst case error can be specified. It can be shown that k-NN performs no worse than twice the Bayes error e_{Bayes} (Theodoridis and Koutroumbas, 2009) for a huge training data set ($N \to \infty$). In order to really quantify the worst case error in practice (1) the Bayes error must be known and (2) it must be known if the size of the training data set is big enough. In addition, since k-NN works on distances, it can be used to classify arbitrary data sets, as long as a distance measure can be defined.

By simply storing the entire training data set, training with k-NN is very fast. Classification on the other hand is quadratic in the number of training instances ($O(kN^2)$ (Theodoridis and Koutroumbas, 2009)). For large data sets, nearest neighbour techniques are still feasible using variants of k-NN that do not require to keep all instances, but rather some representative instances.

Artificial neural networks have probably been the most popular technique in machine learning in the last 30 years. However, the training for neural networks with backpropagation is not straightforward. A disadvantage is that the network structure has to be pre-defined. The gradient descent optimisation has random parts, for example the weights are initialised randomly and the results can depend on the order the instances are presented to the neural network. A further drawback is that there are local minima in the optimisation process. Various classification tasks have been solved with neural networks, however the one-class case is a challenge. In (Moya and Hush, 1996), it was

shown that one-class classification using ANNs is possible, but the proposed approach is rather complex comprising three neural networks.

Finally, support vector machines have the major advantage that the optimisation problem yields the global minimum. The decision function is determined purely mathematically, without random fractions and the classification process is fast. Another advantage is that the decision function is expressed by a small number of support vectors.

In conclusion to this, from the discussed techniques, k-nearest neighbour was selected as a candidate due to its simplicity and good interpretability of the results. Due to the deterministic way of determining the decision function and the compact way of representing it, support vector machines were identified as a further promising candidate.

Chapter 6

One-class classifiers

This chapter compares adaptations of the classifiers k-nearest neighbours and local outlier factor with the one-class classifier support vector data description (SVDD). SVDD is identified to be most suitable for anomaly detection, though suffers from the problem of having to specify parameters beforehand. To solve this problem, a novel parameter tuning approach is proposed, making SVDD applicable to the problem at hand.

In this chapter the multi-class classifier k-NN is thresholded in order to operate as a one-class classifier. Following that, it is discussed how the outlier detection technique local outlier factor (LOF) could be used for one-class classification. Conclusively a one-class support vector machine is enhanced to be applicable to the problem discussed in this Thesis.

6.1 One-class k-NN

This section discusses how k-NN, introduced in Section 5.2.3, could be adapted to the problem of one-class classification.

As a first step, the training takes place analogously to Section 5.2.3 by storing all instances in the knowledge base. In order to classify a test instance x_{test}, the set of the k nearest neighbours $\mathcal{N}(x_{test})$ is determined. Classification in two-class problems would then be based on the class labels in $\mathcal{N}(x_{test})$. However, in one-class problems there is only one class label. The only information available is, how close x_{test} lies to the instances in the training set, i.e. the k distances to $\mathcal{N}(x_{test})$.

As a score of how far apart a test instance is from its k nearest neighbours $\mathcal{N}(x_{test})$, the maximum of the k distances could be used, i.e. the distance to its most remote neighbour in $\mathcal{N}(x_{test})$. However, the reason to set $k > 1$ is to limit the influence of one individual instance. So in terms of determining the threshold this would turn k-NN into 1-NN. So for each x_{test}, the averaged distance to the k neighbours is used, which is given by

$$dist_{avg}(x_{test}) = \frac{1}{k} \sum_{x_i \in \mathcal{N}(x_{test})} dist(x_{test}, x_i) \tag{6.1}$$

Based on $dist_{avg}(x_{test})$, the test instances can be scored and ranked. Autonomous classification is not straightforward though. Since the value range of the distances depends on the data set, it is not possible for a user to pre-define a threshold for an abnormal instance.

6.1.1 Thresholding k-NN

The idea is to determine the threshold in the training period from the inner distances of the training set. For a training data set with M instances, M distances $dist_{avg}$ are obtained and stored in the vector of inner distances D.

The threshold should be determined based on the following trade-off:

1. prevent the threshold from being too small, because that would classify instances that fall into sparser regions within the cluster as abnormal, i.e. would introduce holes into the clusters

2. prevent the threshold from being too large, since the threshold forms a boundary based on the radii around the cluster's outmost instances

It is proposed to determine the threshold from the entire training set. From the vector of distances D, the maximum inner distance $max(D)$ is used as the threshold.

$$threshold_{kNN} = max(dist_{avg_i} \forall i) \tag{6.2}$$

Independent of the threshold's value, an overestimation of the normal class is systematic, since the decision boundary is given by the radius $threshold_{kNN}$ around the outmost instances. Hence, the classifier has a bias towards the normal class, so it is likely to have an unnecessarily high false positive rate. This is depicted in Figure 6.1 for 1-NN. For an anomaly detection system, initialising the threshold with $threshold_{kNN}$ and having the user adjust the threshold over the runtime of the system, would be a good approach.

6.1.2 Discussion

Due to the way the threshold is determined, the training period is computationally expensive for huge data sets. The threshold can be regarded as global, because it is independent of where x_{test} lies in feature space. This is only a good approach if the training set contains clusters with equal densities.

To overcome this problem it is necessary to not solely rely on the distance from x_{test} to its nearest neighbours, but incorporate the neighbours of the nearest neighbours. This has been used in (de Ridder et al., 1998) for 1-NN, where the ratio of the distance

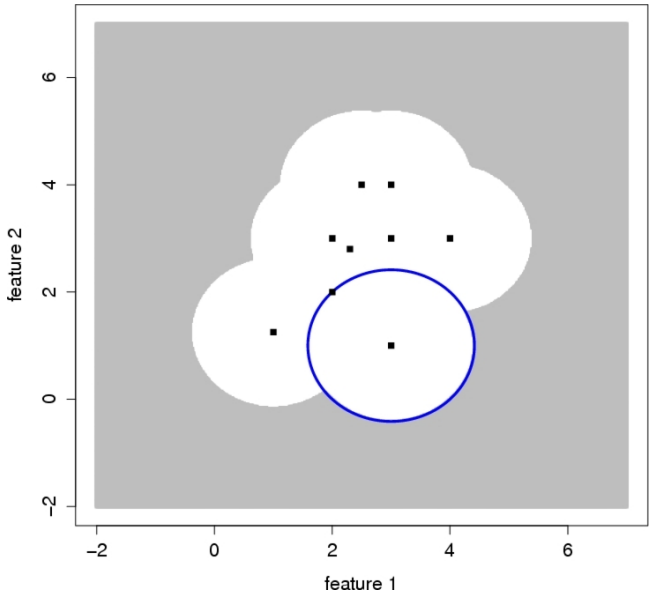


Figure 6.1: Adaptation of k-NN to function as a one-class classifier by determining a threshold from the training set using the maximum nearest neighbours distance (blue circle). The region of the abnormal class is depicted in grey.

from x_{test} to its nearest neighbour $\mathcal{N}(x_{test})$ and the distance between $\mathcal{N}(x_{test})$ and its nearest neighbour is used.

6.2 Local outlier factor

Distance-based approaches like k-NN rely on the distance between an instance x_{test} and its k nearest neighbours. If the distance $dist(x_{test})$ is greater than some global threshold, x_{test} is classified as abnormal.

In Figure 6.2 an example is illustrated. k-NN solely relies on distances and applies the same global threshold to x_{test_1} and x_{test_2}. Hence, either both or none are classified as anomaly. However, x_{test_1} is likely to be an anomaly, while x_{test_2} could belong to the neighbouring cluster. Intuitively it is desired that an instance is classified as abnormal, if it is slightly outside a dense cluster or far outside a sparse cluster.

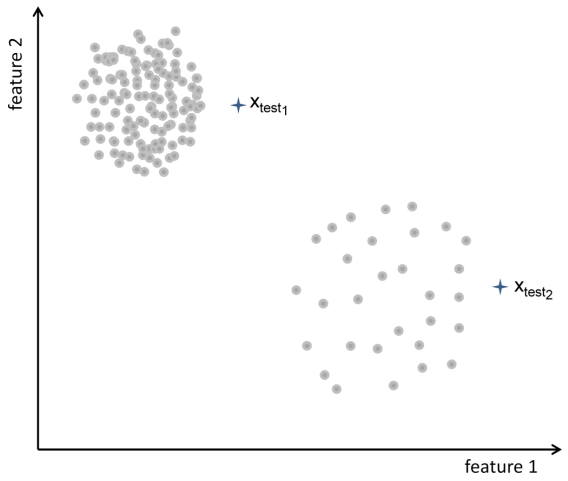


Figure 6.2: Data set with two clusters with unequal densities.

To achieve this, the local outlier factor (LOF) was proposed in (Breunig et al., 2000). LOF incorporates the so-called local density in addition to the neighbourhood. The local density of x_{test} is compared with the local densities of its nearest neighbours. This way LOF works well for data sets with several clusters with unequal densities, while k-NN does not.

6.2.1 Functioning of LOF

As a first step, the k nearest neighbours are determined. The distance to the k^{th} nearest neighbour is referred to as the k-distance $dist_k$ and the set of nearest neighbours is denoted by $\mathcal{N}(x_{test})$. If more than one instance is $dist_k$ away from x_{test}, $\mathcal{N}(x_{test})$ has more than k instances.

Based on $dist_k$, the reachability distance $rdist(x_{test}, x_i)$ is defined in accordance with (Breunig et al., 2000), where x_i is one instance in the set of nearest neighbours:

$$rdist(x_{test}, x_i) = \max \{dist_k(x_i), dist(x_{test}, x_i)\} \tag{6.3}$$

For instances far apart, $rdist(x_{test})$ is the distance between x_{test} and x_i, while for instances close to each other, it is the k-distance. The reachability distance, averaged over all nearest neighbours in $\mathcal{N}(x_{test})$ is given by the following equation, where $\|\mathcal{N}(x_{test})\|$ is the number of nearest neighbours:

$$rdist_{avg}(x_{test}) = \frac{\sum\limits_{x_i \in \mathcal{N}(x_{test})} rdist(x_{test}, x_i)}{\|\mathcal{N}(x_{test})\|} \tag{6.4}$$

Based on this, the reachability density $rdens(x_{test})$ is defined, which is the inverse of the averaged reachability distance, i.e. $\frac{1}{rdist_{avg}}$

$$rdens(x_{test}) = \frac{\|\mathcal{N}(x_{test})\|}{\sum\limits_{x_i \in \mathcal{N}(x_{test})} rdist(x_{test}, x_i)} \tag{6.5}$$

Finally, the local outlier factor for x_{test} is determined by summing up the ratios of the density of each neighbouring instance in $\mathcal{N}(x_{test})$ and the density of x_{test}, normalised by the number of neighbouring instances (Breunig et al., 2000).

$$LOF(x_{test}) = \frac{\sum_{x_i \in \mathcal{N}(x_{test})} \frac{rdens(x_i)}{rdens(x_{test})}}{\|\mathcal{N}(x_{test})\|} \tag{6.6}$$

6.2.2 Thresholding LOF

Assuming equal densities for x_{test} and all instances in $\mathcal{N}(x_{test})$, the LOF becomes 1. If the densities of the neighbouring instances are higher than the density of x_{test}, the LOF is > 1. Hence, for an LOF $\gg 1$, an instance can be regarded as an anomaly. However, there is no absolute LOF value that could be used as a threshold, since the values depend on the data set.

For the experiments in this Thesis, the maximum LOF value within the training set is used as the threshold.

$$threshold_{LOF} = max(LOF(x_i) \forall i) \tag{6.7}$$

Determining the threshold is computationally expensive, since the LOF for each instance has to be calculated.

6.3 Support vector data description

While two-class support vector machines separate the data by a hyperplane (Theodoridis and Koutroumbas, 2009; Abe, 2010), in (Tax and Duin, 1999) support vector data description[1] (SVDD) was introduced to cope with the problem of one-class classification. While in publications (Tax and Duin, 1999, 2004), the fundamentals of SVDD are given in a condensed form, this chapter gives a more thorough insight into SVDD.

[1] When SVDD was introduced in (Tax and Duin, 1999) it was termed support vector domain description, but in later publications (Tax, 2001; Tax and Duin, 2004) renamed to support vector data description.

SVDD finds a closed decision boundary around the normal instances in the training data set. Therefor SVDD uses a hypersphere, which is a sphere generalised to multi-dimensional space, corresponding to a circle in two and a sphere in three dimensions respectively. The hypersphere is fully determined by its radius R and its center a, as illustrated in Figure 6.3, and is found by solving the optimisation problem of minimising:

1. the error on the normal class (false negatives)

2. the chance of misclassifying data from the abnormal class (false positives)

Minimising the error on the normal class is achieved by adjusting R and a in a way that all instances of the training data set are contained in the hypersphere. On the other hand, minimising the chance of misclassifying data from the abnormal class cannot be achieved straightforward, since in the absence of abnormal training data, false positives cannot be measured during the optimisation step.

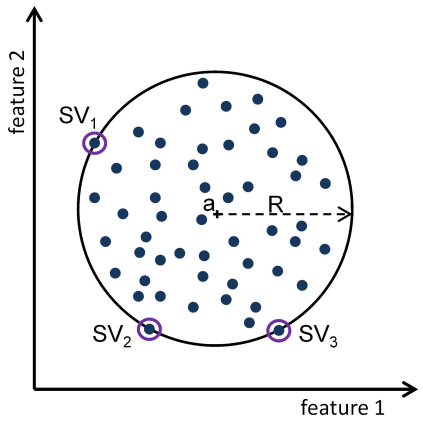


Figure 6.3: A hypersphere in a 2-dimensional feature space with radius R and center a, described by the three support vectors $SV_1 \cdots SV_3$.

6.3.1 Finding the optimal hypersphere

A hypersphere with an infinite volume would obviously enclose all instances but misclassify all abnormal instances. So the hypersphere's volume is used as a second

optimisation criterion. The trade-off between the number of misclassified normal instances and the volume of the normal region is optimised. On the one hand the decision boundary is desired to capture the normal instances, while on the other hand keeping the hypersphere's volume at a minimum. This follows (Moya and Hush, 1996), where minimising the error and the volume of the size of the decision boundary is proposed.

While the volume of a 3-dimensional sphere is given by $V_{sphere} = \frac{4}{3}\pi R^3$, determining the volume of a hypersphere in more than three dimensions is not straightforward. Minimising the hypersphere's volume is equivalent to minimising the area A of the hypersphere's 2-dimensional projection, which corresponds to a circle given by

$$A_{circle} = \pi R^2 \tag{6.8}$$

Since π is irrelevant for the optimisation problem, this reduces to determining the hypersphere's radius and center such that R^2 is minimised. In addition it is demanded that all instances of the training data set are contained in the hypersphere. Hence, the following optimisation problem is to be solved:

minimise

$$F(R, a) = R^2 \tag{6.9}$$

subject to

$$\|x_i - a\|^2 \leq R^2 \quad \forall i \qquad i = 1, .., M \tag{6.10}$$

where x_i denotes the instances and M the number of instances in the training data set, a is the hypersphere's center, and $\|x_i - a\|$ is the distance between x_i and a.

The distance $\|x_i - a\|$ is calculated by $\sqrt{(x_i - a) \cdot (x_i - a)}$. Since calculating the square root is computationally expensive, the squared distance $\|x_i - a\|^2$ is used and

compared to the squared radius R^2. The squared distance can be reformulated using the binomial theorem which is beneficial, as will become clear in Section 6.3.4. Hence, in accordance to (Tax and Duin, 1999), the squared distance and the squared radius are used in this Thesis.

The decision boundary, i.e. the hypersphere, is described by selected instances from the training data set, so-called support vectors. The center a is implicitly described by a linear combination of the support vectors. The remaining instances are discarded.

Having solved the optimisation problem eq. (6.9) and eq. (6.10) and hence having found the hypersphere, for each x_i one of two terms is satisfied. This is used to select those x_i that become support vectors:

$$\|x_i - a\|^2 < R^2 \quad \rightarrow \quad x_i \text{ is within the hypersphere, i.e can be discarded} \tag{6.11}$$

$$\|x_i - a\|^2 = R^2 \quad \rightarrow \quad x_i \text{ is selected as a support vector} \tag{6.12}$$

As an illustrative example one could think of a 2-dimensional training data set with 50 instances. In this case the decision boundary would be a circle, which is fully described by three distinct points on its circumference. Hence, ideally three support vectors would be selected and 47 instances be discarded as illustrated in Figure 6.3.

Classifying a test instance x_{test} is a matter of determining whether it is inside or outside the hypersphere, which is done by solving

$$\|x_{test} - a\|^2 \begin{cases} \leq R^2 : & \text{if } x_{test} \text{ is inside the hypersphere, i.e. classified as positive} \\ > R^2 : & \text{if } x_{test} \text{ is outside the hypersphere, i.e. classified as negative} \end{cases} \tag{6.13}$$

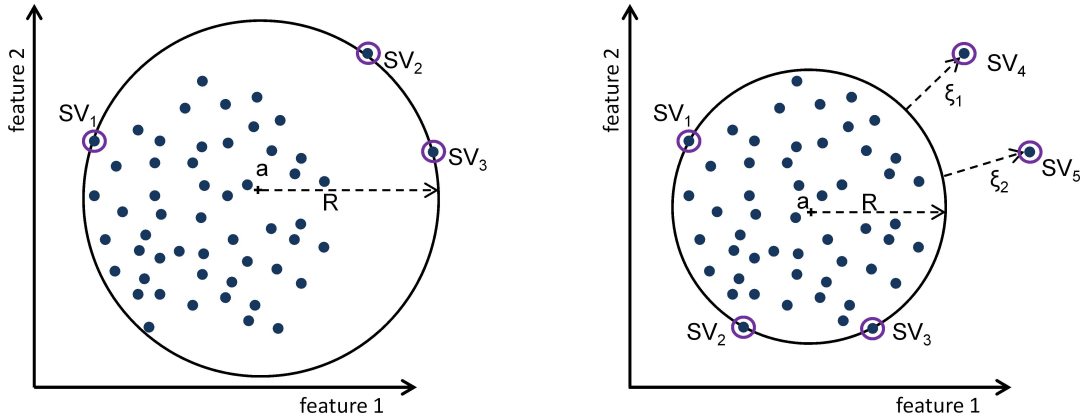


Figure 6.4: The introduction of the slack variables ξ_i allows for some instances of the training data set to be outside the decision boundary. (a) without slack variables (b) with slack variables.

6.3.2 Reducing the sensitivity to outliers

Demanding that *all* instances are contained in the hypersphere means that outliers contained in the training data set will massively influence the decision boundary. So SVDD in this form is very sensitive to outliers, which is not desired.

Analogous to the case of hard-margin SVMs, that can be transformed to soft-margin SVMs by allowing some instances to be on the wrong side of the separating hyperplane (Abe, 2010), in SVDD slack variables are introduced. These slack variables ξ_i, given by eq. (6.14), allow for some instances x_i in the training data set to be outside the hypersphere as shown in Figure 6.4.

$$\xi_i = \begin{cases} \|x_i - a\|^2 - R^2 & : \quad \text{if } (\|x_i - a\|^2 - R^2) > 0 \\ 0 & : \quad \text{if } (\|x_i - a\|^2 - R^2) \leq 0 \end{cases} \tag{6.14}$$

The parameter C is introduced controlling the influence of the slack variables and thereby the error on the normal class and the hypersphere's volume. So the optimisation problem of eq. (6.9) and eq. (6.10) becomes:

minimise

$$\boxed{F(R, a, \xi_i) = R^2 + C \sum_{i=1}^{M} \xi_i} \tag{6.15}$$

subject to

$$\boxed{\|x_i - a\|^2 \leq R^2 + \xi_i \quad \forall i} \tag{6.16}$$

and

$$\boxed{\xi_i \geq 0 \quad \forall i} \tag{6.17}$$

6.3.3 Solving the optimisation problem

As described in (Tax and Duin, 2004), the optimisation problem is solved by incorporating the constraints eq. (6.16) and eq. (6.17) into eq. (6.15) using the method of Lagrange for positive inequality constraints (Jones, 2005). This allows transforming a constrained optimisation problem into an unconstrained one by integrating the constraints into the equation to be optimised. First eq. (6.16) is rewritten to become a positive inequality constraint:

$$R^2 + \xi_i - \|x_i - a\|^2 \geq 0 \tag{6.18}$$

For a function f and two constraints $g_1 \geq b_1$ and $g_2 \geq b_2$, the Lagrangian is formulated as $L = f - \alpha(g_1 - b_1) - \beta(g_2 - b_2)$, introducing the so-called Lagrange multipliers α_i and β_i. For the constraints eq. (6.17) and eq. (6.18) b_1 and b_2 are equal to 0. Incorporating

the constraints eq. (6.17) and eq. (6.18) into eq. (6.15), the optimisation problem changes into minimising

$$\begin{aligned} L(R, a, \alpha, \beta, \xi) &= R^2 + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\alpha_i(R^2 + \xi_i - \|x_i - a\|^2 - 0) - \sum_{i=1}^{M}\beta_i(\xi_i - 0) \\ &= R^2 + C\sum_{i=1}^{M}\xi_i - \sum_{i=1}^{M}\alpha_i(R^2 + \xi_i - x_i^2 + 2ax_i - a^2) - \sum_{i=1}^{M}\beta_i\xi_i \end{aligned} \quad (6.19)$$

In order to find the minimum, the partial derivatives are set to 0. Since it is a convex optimisation problem this uniquely yields the minimum. Setting the partial derivative $\frac{\partial L}{\partial R}$ to 0

$$\frac{\partial L}{\partial R} = 2R - 2R\sum_{i=1}^{M}\alpha_i = 0 \quad (6.20)$$

yields the condition

$$\sum_{i=1}^{M}\alpha_i = 1 \quad (6.21)$$

Setting the partial derivative with respect to a to 0

$$\begin{aligned} \frac{\partial L}{\partial a} &= -2\sum_{i=1}^{M}(\alpha_i x_i - \alpha_i a) = 0 \\ &= \sum_{i=1}^{M}\alpha_i x_i - \sum_{i=1}^{M}\alpha_i a = 0 \end{aligned} \quad (6.22)$$

yields the following condition

$$a = \frac{\sum_{i=1}^{M} \alpha_i x_i}{\sum_{i=1}^{M} \alpha_i} \qquad \text{with } \sum_{i=1}^{M} \alpha_i = 1 \quad \text{from eq. (6.21)}$$
$$a = \sum_{i=1}^{M} \alpha_i x_i \tag{6.23}$$

which shows that the center a is expressed as a linear combination of the support vectors. Finally, deriving with respect to ξ_i leads to

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \tag{6.24}$$

Since $\alpha_i \geq 0$ and $\beta_i \geq 0$, and $\beta_i = C - \alpha_i$ this allows to drop β_i by instead adding the following constraint

$$0 \leq \alpha_i \leq C \tag{6.25}$$

Assuming that eq. (6.19) is minimal, resubstituting the found constraints yields a less complex equation. First eq. (6.19) is reformulated as

$$\begin{aligned} L = & R^2 + C \sum_{i=1}^{M} \xi_i - \sum_{i=1}^{M} \alpha_i R^2 - \sum_{i=1}^{M} \alpha_i \xi_i \\ & + \sum_{i=1}^{M} \alpha_i x_i^2 - 2 \sum_{i=1}^{M} \alpha_i a x_i + \sum_{i=1}^{M} \alpha_i a^2 - \sum_{i=1}^{M} \beta_i \xi_i \end{aligned} \tag{6.26}$$

Substituting $(\sum_{i=1}^{M} \alpha_i) = 1$ from eq. (6.21) shows that the terms containing R^2 cancel each other out

$$L = C\sum_{i=1}^{M} \xi_i - \sum_{i=1}^{M} \alpha_i \xi_i + \sum_{i=1}^{M} \alpha_i x_i^2 - 2\sum_{i=1}^{M} \alpha_i a x_i + \sum_{i=1}^{M} \alpha_i a^2 - \sum_{i=1}^{M} \beta_i \xi_i \tag{6.27}$$

and substituting $\beta_i = C - \alpha_i$ from eq. (6.25), allows to drop the terms containing C and ξ_i

$$\begin{aligned} L &= C\sum_{i=1}^{M} \xi_i - \sum_{i=1}^{M} \alpha_i \xi_i + \sum_{i=1}^{M} \alpha_i x_i^2 - 2\sum_{i=1}^{M} \alpha_i a x_i + \sum_{i=1}^{M} \alpha_i a^2 - \sum_{i=1}^{M} (C - \alpha_i)\xi_i \\ &= C\sum_{i=1}^{M} \xi_i - \sum_{i=1}^{M} \alpha_i \xi_i + \sum_{i=1}^{M} \alpha_i x_i^2 - 2\sum_{i=1}^{M} \alpha_i a x_i + \sum_{i=1}^{M} \alpha_i a^2 - C\sum_{i=1}^{M} \xi_i + \sum_{i=1}^{M} \alpha_i \xi_i \\ &= \sum_{i=1}^{M} \alpha_i x_i^2 - 2\sum_{i=1}^{M} \alpha_i a x_i + \sum_{i=1}^{M} \alpha_i a^2 \end{aligned} \tag{6.28}$$

Finally, substituting $a = (\sum_{i=1}^{M} \alpha_i x_i)$ from eq. (6.22), the equation becomes

$$L = \sum_{i=1}^{M} \alpha_i x_i^2 - 2\sum_{i=1}^{M} \alpha_i x_i \sum_{j=1}^{M} \alpha_j x_j + \sum_{i=1}^{M} \alpha_i x_i \sum_{j=1}^{M} \alpha_j x_j \tag{6.29}$$

So the optimisation problem changes into maximising[2]

[2]If the reader refers to the references: In his PhD Thesis (Tax, 2001), David Tax accidently stated that the optimisation problem has to be minimised, while in (Tax and Duin, 1999, 2004) he correctly stated it has to be maximised.

$$\boxed{L(\alpha) = \sum_{i=1}^{M} \alpha_i (x_i \cdot x_i) - \sum_{i,j=1}^{M} \alpha_i \alpha_j (x_i \cdot x_j)} \tag{6.30}$$

with respect to α, subject to

$$\boxed{0 \leq \alpha_i \leq C \quad \forall i} \tag{6.31}$$

Hence, the original problem eq. (6.15) that had to be optimised w.r.t. R, a, and ξ with two constraints, was simplified to optimising w.r.t. α, with box constraints on the α_i. The second term in eq. (6.30) indicates a quadratic form (for $i = j$, $\alpha_i \alpha_j$ becomes α_i^2). This optimisation problem can be solved using quadratic programming. The resulting values for α_i indicate the position of an instance x_i as follows:

$$\alpha_i \quad \begin{cases} = 0 & : x_i \text{ is inside the hypersphere} \\ > 0; < C & : x_i \text{ is on the boundary, } x_i \text{ becomes a support vector} \\ = C & : x_i \text{ is outside the hypersphere, } x_i \text{ becomes a support vector} \end{cases} \tag{6.32}$$

Having determined all α_i, the parameters a and ξ_i can be deduced from the α's. The radius R is determined by picking an arbitrary support vector x_i on the boundary, i.e. with $0 < \alpha_i < C$, and solving $R = \|x_i - a\|$.

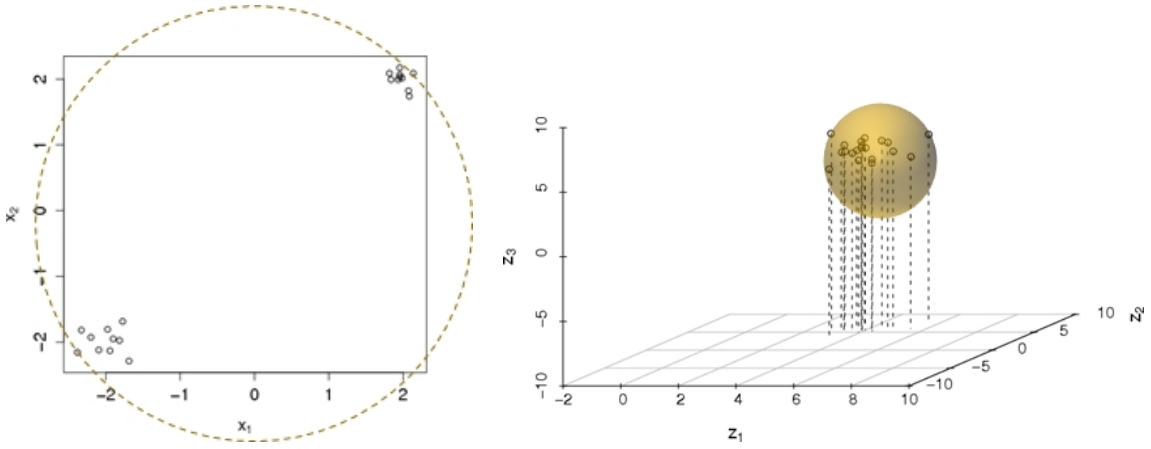
6.3.4 Introducing non-spherical decision boundaries

At this point, SVDD is capable of surrounding the normal data by a hypersphere. It is rare that in classification problems the distribution of the data is spherical. In the

problem presented in this Thesis it is not realistic. Hence, SVDD in this form would yield poor classification results.

Analogous to the traditional two-class support vector machines, SVDD uses the so-called kernel trick (Theodoridis and Koutroumbas, 2009) to overcome this shortcoming. The two-class support vector machines linearly separate the data by a hyperplane which in most classification problems yields high error rates. Using a kernel function the data is mapped to a higher-dimensional space, where it can be linearly separated as described in Section 5.2.3.4. Using kernels known from SVMs, SVDD maps the training data set to a higher-dimensional space where it can be surrounded by a hypersphere.

The contrived two-dimensional data set in Figure 6.5(a) has instances from the normal class distributed in two clusters. Enclosing all instances with a sphere would massively overestimate the normal class. Using the polynomial mapping from Section 5.2.3.4 as an example, the data set can be mapped to a three dimensional space, as shown in Figure 6.5(b), where the instances can be surrounded by a sphere.



(a) Instances from the normal class distributed in two clusters. Enclosing all instances with a sphere would massively overestimate the normal class.

(b) The instances of the contrived data set can be enclosed by a sphere in the three-dimensional feature space created by the mapping function.

Figure 6.5: Example of how a data set can be enclosed by a sphere by mapping it from two- to three-dimensional feature space.

As can be seen from eq. (6.30) x_i and x_j are solely incorporated as the inner products (scalar products) $(x_i \cdot x_i)$ and $(x_i \cdot x_j)$ respectively. Instead of actually mapping each instance to a higher-dimensional space using a mapping function $\phi()$

$$L(\alpha) = \sum_{i=1}^{M} \alpha_i(\phi(x_i) \cdot \phi(x_i)) - \sum_{i,j=1}^{M} \alpha_i \alpha_j(\phi(x_i) \cdot \phi(x_j)) \tag{6.33}$$

the kernel trick is to replace the inner products $(\phi(x_i) \cdot \phi(x_j))$ by a kernel function $K(x_i, x_j)$. The mapping is implicitly done by solving $K(x_i, x_j)$.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{6.34}$$

So eq. (6.30) becomes:

$$L(\alpha) = \sum_{i=1}^{M} \alpha_i K(x_i, x_i) - \sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i, x_j) \tag{6.35}$$

A variety of kernel functions have been proposed. Two widely used kernels are the polynomial kernel, where d denotes the polynomial's degree

$$K(x_i, x_j) = (x_i \cdot x_j)^d \tag{6.36}$$

and the Gaussian kernel, also referred to as the radial basis function (RBF) kernel

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}} \tag{6.37}$$

where σ is referred to as the kernel width. In some publications the kernel parameter is denoted as γ, where $\gamma = \frac{1}{\sigma^2}$ and eq. (6.37) becomes $e^{-\gamma\|x_i - x_j\|^2}$.

6.3.4.1 Classifying a test instance

In the classification step, a test instance x_{test} is classified by solving

$$\|x_{test} - a\|^2 > R^2 \tag{6.38}$$

The squared distance $\|x_{test} - a\|^2$ can be rewritten as

$$x_{test} \cdot x_{test} - 2x_{test} \cdot a + a \cdot a \tag{6.39}$$

Replacing a by its linear combination of support vectors from eq. (6.23) yields

$$x_{test} \cdot x_{test} - 2\sum_{i=1}^{M} \alpha_i (x_{test} \cdot x_i) + \sum_{i,j=1}^{M} \alpha_i \alpha_j (x_i \cdot x_j) \tag{6.40}$$

Again, the inner products are replaced by the kernel function used during training. A test instance is classified as abnormal, if the following inequality holds.

$$K(x_{test}, x_{test}) - 2\sum_{i=1}^{M} \alpha_i K(x_{test}, x_i) + \sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i, x_j) > R^2 \tag{6.41}$$

As can be seen from eq. (6.41), classification is very fast compared to k-NN and LOF. It involves basic vector algebra and the application of the kernel function on the support vectors x_i.

6.3.4.2 The RBF kernel

The Gaussian kernel is reported to be most suitable to be used with SVDD in (Tax and Duin, 2004). As opposed to the polynomial kernel, the Gaussian kernel does not depend on the position of instances with respect to the origin (Tax and Duin, 2004). A major advantage for the problem discussed in this Thesis is, that this kernel adds only one parameter to the classification problem, the kernel width σ.

The essential integral part of SVDD is the use of the kernel trick. The formally correct explanation that the data set is mapped to a higher-dimensional space without actually conducting the mapping might be incomprehensible at first sight. An alternative to gain insight into the kernel trick is as follows. In the case of a Gaussian kernel, another way is to think of kernel functions as distance functions, where the influence degrades according to a Gaussian function the more distant two instances x_i and x_j are. Since this is done for all x_i, this results in a non-linear mapping.

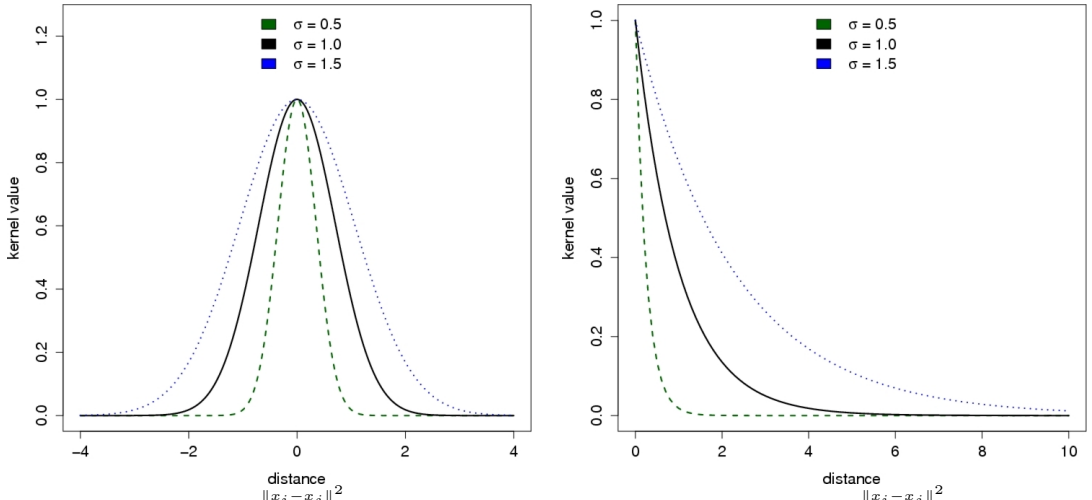
The kernel function $e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$ can take on values from the interval $(0, 1]$ and converges to 0 for big distances $\|x_i - x_j\|$. The influence on neighbouring feature vectors for different values of σ is shown in Figure 6.6(a). For a growing distance the function is illustrated for different values of σ in Figure 6.6(b).

Since $K(x_i, x_i) = e^{-\frac{\|x_i - x_i\|^2}{\sigma^2}} = 1$ and $\sum_{i=1}^{M} \alpha_i = 1$ the previous equations can be simplified for the RBF kernel. The optimisation problem from eq. (6.35) to find the optimal hypersphere is then given by:

maximise

$$L(\alpha) = 1 - \sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i \cdot x_j) \tag{6.42}$$

subject to



(a) Kernel function $e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$ for different values of σ.

(b) The kernel function $e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$ converges to 0.

Figure 6.6: RBF kernel for different values of σ.

$$0 \leq \alpha_i \leq C \quad \forall i \tag{6.43}$$

The equation to classify an instance eq. (6.41) boils down to:

$$1 - 2\sum_{i=1}^{M} \alpha_i K(x_{test}, x_i) + \sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i, x_j) > R^2 \tag{6.44}$$

$$1 - 2\sum_{i=1}^{M} \alpha_i e^{-\frac{\|x_{test} - x_i\|^2}{\sigma^2}} + \sum_{i,j=1}^{M} \alpha_i \alpha_j e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}} > R^2 \tag{6.45}$$

As an example, for three feature vectors $x_1 \ldots x_3$ the center a shall be given by a linear combination of the three vectors as illustrated in Figure 6.7 by

$$a = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 \tag{6.46}$$

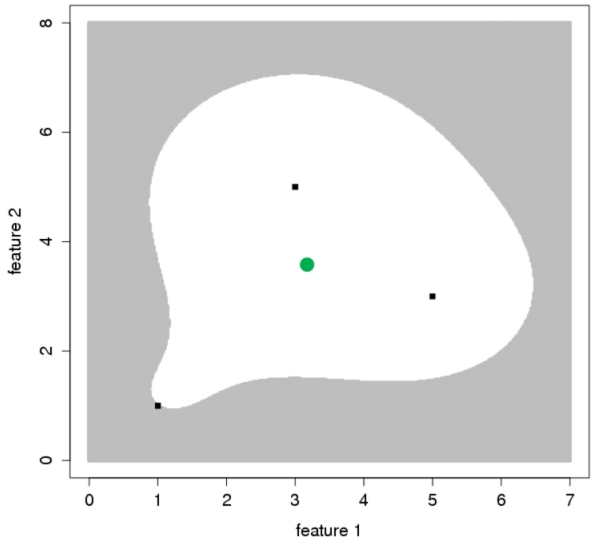


Figure 6.7: Non-linear decision function using the RBF kernel for three feature vectors and the center a (green) described as the linear combination of the feature vectors.

Now calculating the squared distance from the center $\|x_i - a\|^2$ for any point in the shown feature space and comparing the result to R^2 yields the decision boundary shown in Figure 6.7. As can be seen, the decision function is non-linear.

6.3.5 Surveying ways to determine the SVDD parameters

The optimisation problem eq. (6.15) is solved for a given set of parameters C and σ. Determining those parameters is not straightforward. This section discusses ways of finding the optimal parameter set.

The first parameter is the regularisation parameter C introduced in eq. (6.15). C is lower-bound by $\frac{1}{N}$, where N is the number of instances in the training data set. $C = 1$ corresponds to the hard-margin solution, where all instances are enclosed in the decision boundary (Tax and Duin, 1999). So the value range of C is

$$\frac{1}{N} \leq C \leq 1 \tag{6.47}$$

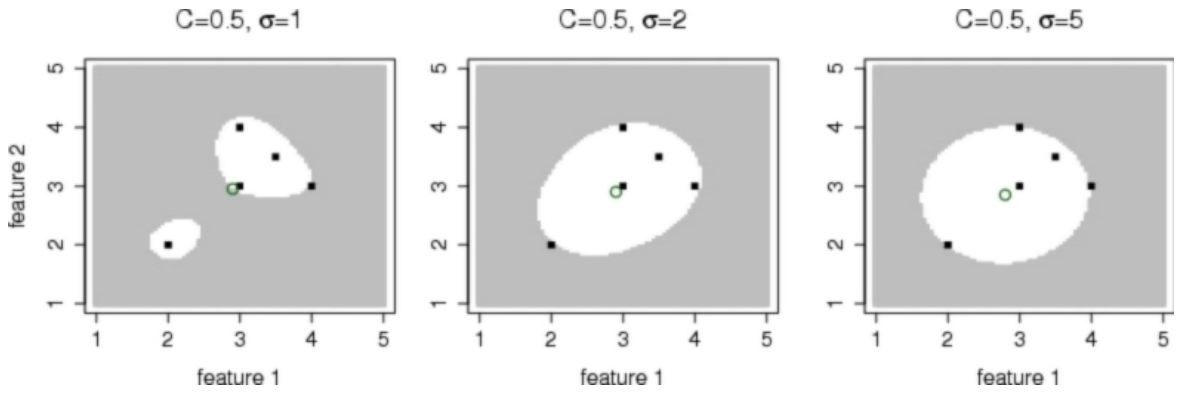


Figure 6.8: The influence of the parameter σ on the decision boundary. The black squares are the feature vectors and the circle is the determined center.

In this Thesis the RBF kernel is used, which adds the second parameter to be optimised, the kernel width σ as given in eq. (6.37). For high values of σ the boundary will become spherical with the risk of underfitting, while for small values of σ a high fraction of instances are selected to be support vectors, hence the boundary is very flexible and is prone to overfitting. The influence of σ on the kernel function is shown for a small contrived data set in Figure 6.8.

In two-class classification problems, the classifier's parameters are optimised using cross-validation or leave-one-out validation based on a labelled training data set containing both classes. For a given set of parameters, true positives, false positives, true negatives, and false negatives are measured and listed in the so-called confusion matrix. Based on the measured values the classifier's parameters are optimised w.r.t. for example the overall-accuracy, the true positive rate, or further metrics (Fawcett, 2004) depending on the application.

Since in one-class classification problems only instances from the normal class are contained in the training data set, only true positives and false negatives can be measured.

Several approaches to find the parameters for SVDD have been proposed. In (Tax and Duin, 2004) hints are given about determining parameter C. If there is certainty that no outliers exist in the training data set, C can be set to 1, so that the hypersphere

will include all instances. If knowledge about the fraction of outliers ν in the training data set exists, the parameter should be set to $C \leq \frac{1}{\nu N}$.

In (Zhuang and Dai, 2006), the authors discuss optimising parameters for SVDD in the context of text classification, where an imbalanced classification problem is encountered. The normal class is much smaller sampled than the abnormal class. SVDD is used to learn from the normal class and then optimised using instances from the abnormal class (Zhuang and Dai, 2006). In (Mack and Waske, 2011) SVDD is used for image classification, stating that in an image there are labelled instances (pixels) that belong to the normal class and there is a huge amount of unlabelled instances. An approach is proposed to select potential abnormal instances from the unlabelled instances to be used for parameter optimisation. Neither (Zhuang and Dai, 2006) nor (Mack and Waske, 2011) address the problem of non-available outliers, but rather propose how to transform the problem to two-class classification for parameter tuning.

In the absence of outlier data, (Tax and Duin, 2001b) proposes to generate artificial instances that are uniformly distributed in a hypersphere around the target class including the region of the target class. Based on the fraction of instances classified as normal an optimisation criterion is defined.

In (Tax and Duin, 2001a) an error function for the Gaussian kernel is defined utilising the number of support vectors, that is used to optimise the two parameters without the need to select or generate outliers. However, it has two drawbacks for the problem discussed in this Thesis: a trade-off parameter is introduced and the results are reported to be far from satisfactory.

In (Tax and Duin, 2004) an estimation of the error on the target set is given, based on the number of essential support vectors. Identifying the essential support vectors is done using leave-one-out. Leaving out an essential support vector from the training set, the decision boundary changes, while it remains unchanged if a non-essential support vector or a normal instance is left out. However, leave-one-out is not practical for large training sets due to its high execution time, e.g. for 10000 instances, 10000 training runs are required.

6.3.6 Autonomously tuning the SVDD parameters

Like most classifiers, SVDD has parameters that massively influence the classification accuracy. While SVDD yields good classification results for the one-class problem, manually adjusting the parameters C and σ makes it non-applicable for the problem discussed in this Thesis.

As surveyed in Section 6.3.5, current approaches either work with available or generated data sets or by heuristically or experimentally setting the parameters. The autonomous approach proposed in (Tax and Duin, 2001a) is reported to yield poor decision boundaries and the approach in (Tax and Duin, 2004) is infeasible for large data sets. This section presents an approach to autonomously tune the parameters by solely working on the training set.

As proposed in (Hsu et al., 2003) for two-class SVMs, grid search is utilised for the selection of parameter candidates. Based on an optimisation criterion the best pair $\{C_i, \sigma\}$ is determined.

The task is to tune the parameters so that the accuracy on the test set and on unseen data is optimal. The input variables of one optimisation step are the training data set $\mathcal{A}$ and the SVDD parameter candidates C_i, σ_i. Relevant measurable output variables are the error rate e_{ω_n}, the radius R, and the number of support vectors.

As described in Section 6.3.4, SVDD transforms the feature space $\mathcal{F}$ into the transformed feature space $\phi(\mathcal{F})$ and then tries to find the optimal hypersphere surrounding the instances in $\phi(\mathcal{F})$.

In order to equally weigh all incorporated features, prior to finding the hypersphere, all features in $\mathcal{F}$ are independently normalised to a value range of $[-1, 1]$ by min/max normalisation.

6.3.6.1 Selecting parameter candidates using grid search

The SVDD parameters C and σ are optimised using grid search in this Thesis. Within a given value range, the grid search algorithm selects candidate values and tests all candidates. The selection of candidate values is done iteratively. While this becomes computationally expensive for many parameters, it is feasible for the two parameters in the problem presented.

For SVDD as used in this Thesis, the required input parameters for grid search are C_{min}, C_{max}, σ_{min}, σ_{max}, the number of candidates in the current range denoted by τ and some stopping criterion like the number of iterations i.

For the parameter C, τ values are selected within the range of $[C_{min}; C_{max}]$, and for σ respectively. This sums up to τ^2 pairs $\{C_i, \sigma_i\}$ as shown in Figure 6.9a. For all $\{C_i, \sigma_i\}$, SVDD is trained and one optimal parameter set $\{C_{opt1}, \sigma_{opt1}\}$ is found.

This is refined in a second iteration by again selecting τ^2 pairs $\{C_i, \sigma_i\}$ in the range of $[C_{opt1-1}; C_{opt1+1}]$ and $[\sigma_{opt1-1}; \sigma_{opt1+1}]$ as shown in Figure 6.9b.

This process is repeated until some stopping criterion is met, e.g. until the value of some optimisation criterion does not further decrease in an iteration or the desired number of iterations is reached. Whether the parameter set has improved, is determined by assessing some optimisation criterion. Therefore it is crucial to identify a good optimisation criterion.

6.3.6.2 Possible optimisation criteria

Obviously a low error rate is desired, so selecting $\{C_i, \sigma_i\}$ where e_{ω_n} is minimal could be a good approach. However, as can be seen from experimental results, this approach overestimates the region of the normal class by selecting too few support vectors (see Figure 6.10). As a consequence the learnt decision boundary does not generalise well.

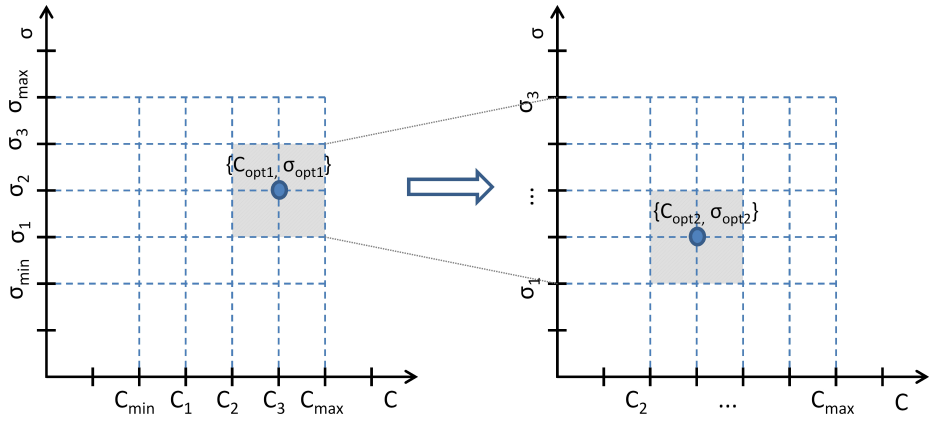
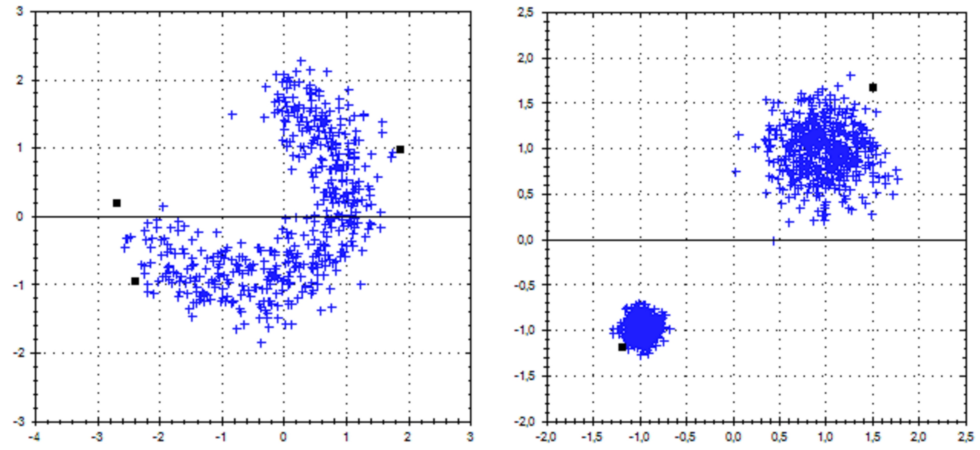


Figure 6.9: Functioning of grid search to optimise the SVDD parameter C and σ with $\tau = 5$ and linear partitioning of the ranges. (a) first iteration (b) second iteration



(a) "banana" data set with only 3 support vectors (black squares: support vectors).

(b) "2 clusters" data set with two support vectors (black squares: support vectors).

Figure 6.10: Results of parameter tuning by minimising the error rate visualised in input feature space, where the selected support vectors do not tightly enclose the training set.

A solution could be to minimise e_{ω_n}, while at the same time maximising the number of support vectors. But, as reported in (Tax, 2001) and confirmed by experiments, the error rate and the number of support vectors are approximately linearly correlated. Hence, optimising the trade-off between e_{ω_n} and the number of support vectors is not possible.

6.3.6.3 Proposed optimisation criterion

The parameters C and σ are optimal if in the transformed feature space, the instances are arranged in a spherical way. Only then will a hypersphere be the ideal decision boundary. So the idea is, to find an optimisation criterion which selects that pair of parameters that best map the data set to one spherical cluster in $\phi(\mathcal{F})$.

Following the kernel trick discussed in Section 6.3.4, the mapping $\mathcal{F} \mapsto \phi(\mathcal{F})$ is not explicitely conducted, it is just an implicit mapping. So how can one determine if the mapping is ideal, without actually conducting the mapping?

The data in the transformed feature space $\phi(\mathcal{F})$ cannot be visualised because (1) it is high-dimensional and (2) it is an implicit mapping. In order to make the point clear, assume that $\phi(\mathcal{F})$ is a known two-dimensional feature space, so that it can be visualised.

The hypersphere's radius can be determined by selecting an arbitrary support vector on the boundary x_b, where x_b can be any x_i for which $0 < \alpha_i < C$ holds. The radius is then the distance between x_b and the center a

$$R^2 = \|x_b - a\|^2 \tag{6.48}$$

which after applying the binomial theorem and incorporating the RBF kernel is given by

$$R^2 = 1 - 2\sum_{i=1}^{M} \alpha_i K(x_b, x_i) + \sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i, x_j) \tag{6.49}$$

From eq. (6.49) two observations are made:

1. The third term in eq. (6.49) is exactly the term that is minimised in the optimisation problem eq. (6.42). The geometric interpretation in the original space is that $a \cdot a$ is minimised when $\|a\|$ is minimised, which is the case when the center a is located at the origin $(0, 0, \dots)$.

2. The second term $\sum_{i=1}^{M} \alpha_i K(x_b, x_i)$ in eq. (6.49) incorporates one selected support vector x_b on the left side of the kernel function and all support vectors x_i on the right side, which is a subset of the third term $\sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i, x_j)$, where all support vectors are incorporated on both sides of the kernel function.

From the second observation, the assumption is deduced, that the second and third terms in eq. (6.49) are proportional. In Figure 6.11 the values of term3 w.r.t. 0.5*term2 are depicted for the entire range of C and σ for four data sets. Experiments have confirmed that the assumption of term2 and term3 being proportional holds for all tested data sets over the entire range of parameters, they in fact are linearly related.

Substituting the 3rd term in eq. (6.49) by b

$$b := \sum_{i,j=1}^{M} \alpha_i \alpha_j K(x_i, x_j) \tag{6.50}$$

and expressing the 2nd term with a proportionality factor c

$$2\sum_{i=1}^{M} \alpha_i K(x_b, x_i) = bc \tag{6.51}$$

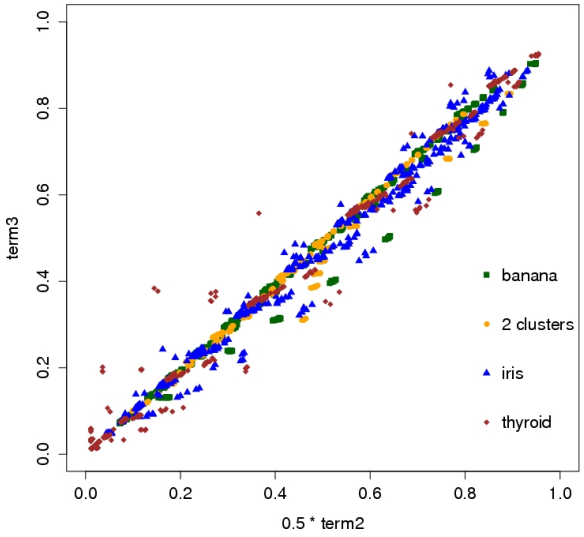


Figure 6.11: The second and third term are proportional for all tested data sets over the entire range of the parameters C and σ.

simplifies eq. (6.49) into following equation

$$R^2 = 1 - bc + b \tag{6.52}$$

which can be rewritten as

$$R = \sqrt{1 - b(c-1)} \tag{6.53}$$

From eq. (6.53) it can be seen that the smaller b is, the closer the radius is to 1. Since b is exactly the term to be minimised, a radius of $R \to 1$ can be considered optimal. Figure 6.12 shows the optimal solution in terms of the radius: the center is at $(0, 0)$ and the radius at 1, i.e. the decision boundary is a unit sphere. Hence, for parameter tuning a radius of 1 is considered as the optimal radius. This finding is based on the observations in the bachelor thesis (Pavlichenko, 2011), which was supervised by the author.

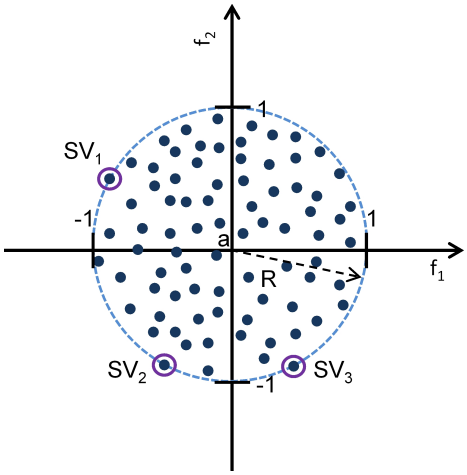


Figure 6.12: Optimal mapping in a constructed transformed feature space. The instances are arranged in a spherical way.

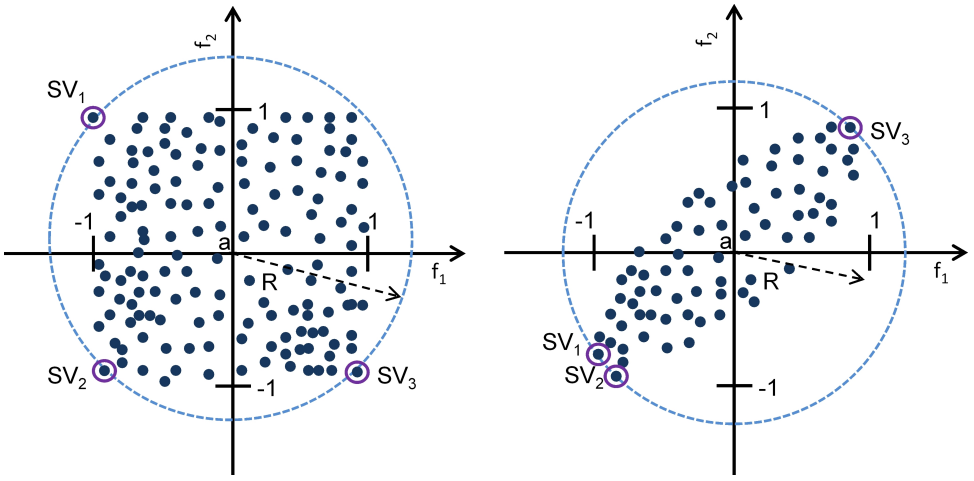
From the kernel function $e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$ it can be seen that the smaller σ is, the closer R will be to 1. So small values of σ are favoured when optimising R. However, for small values of σ, very flexible decision boundaries are obtained as was shown in Figure 6.8, i.e. a high number of support vectors is selected. This tends to overfit the training set, which in turn yields a high error rate. So only optimising for the radius is insufficient.

In order to find $\{C_{opt}, \sigma_{opt}\}$ (see Figure 6.9), the following optimisation criterion is formulated. Informally spoken, R is desired to be close to 1, while at the same time e_{ω_n} is to be minimised.

Finding R close to 1 is equivalent to minimising $|1 - R|$. Equally weighting error rate and radius this boils down to finding the pair $\{e_{\omega_n i}, R_i\}$ closest to the origin by minimising:

$$\lambda_i = \sqrt{{e_{\omega_n}}_i^2 + |1 - R_i|^2} \quad \forall i \tag{6.54}$$

This way, non-optimal mappings are assigned larger distances by eq. (6.54), because non-spherical shapes in $\phi(\mathcal{F})$ are likely to result in a surrounding sphere with $R \neq 1$, as illustrated in Figure 6.13.



(a) Instances in mapped feature space arranged in a square.

(b) Instances in mapped feature space arranged in an ellipse.

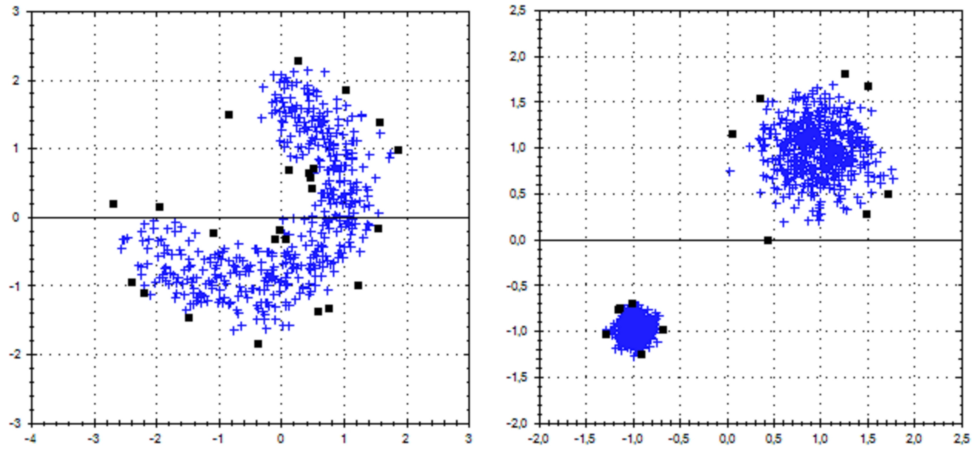
Figure 6.13: Two constructed examples of non-optimal mappings resulting in spheres with $R > 1$.

For each optimisation step, the error rate e_{ω_n} and the radius R are determined using k-fold. The training set is randomly split into k folds. The instances are classified k times with each fold used as the validation set once and e_{ω_n} and R are averaged over the k runs.

Figure 6.14 shows results based on the proposed optimisation criterion. For two artificial two-dimensional data sets, the support vectors that are autonomously selected enclose the training set in the input feature space $\mathcal{F}$. In the case of more than one cluster in input space, with an ideal mapping, SVDD maps all instances to one spherical-shaped cluster in $\phi(\mathcal{F})$, as shown for the two clusters in Figure 6.14b.

6.3.6.4 Functioning of the tuning algorithm

To show the functioning of the tuning algorithm, the data sets in Table 6.1 were used, where $\|\mathcal{F}\|$ is the number of features and $\|\mathcal{A}\|$ is the size of the training set.



(a) Banana-shaped cluster (black squares: support vectors).

(b) Two circular clusters following a Gaussian distribution with different densities (black squares: support vectors).

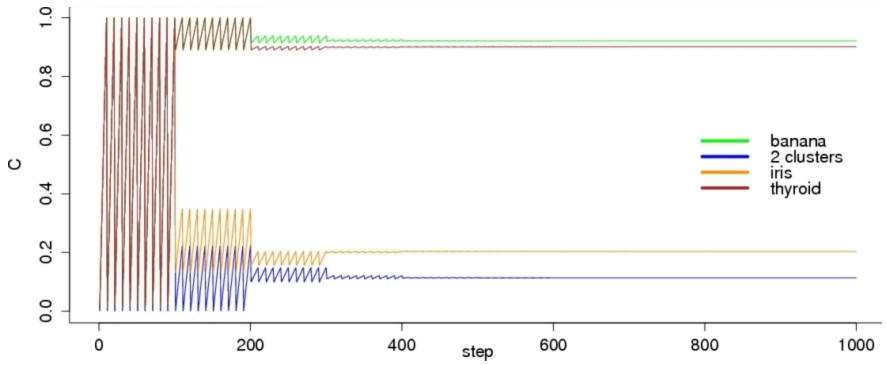
Figure 6.14: Parameter tuning of SVDD on artificial two-dimensional data sets visualised in input feature space.

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$
banana	2	490
2 clusters	2	1000
Iris	4	25
thyroid	21	100

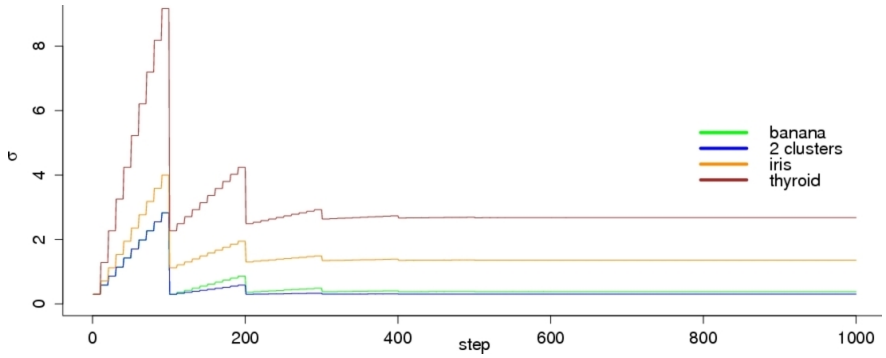
Table 6.1: Properties of data sets used for parameter tuning.

Using grid search, the parameter range was linearly split into 20 candidates, i.e. for the two parameters a grid of 400 (20×20) candidate pairs $\{C_i, \sigma_i\}$ was selected per iteration. The parameter ranges were refined by 10 iterations, summing up to 4000 steps. Within each step k-fold is conducted with k = 7. Experiments have shown that such a high number of steps is far from being necessary, it was chosen to have a fine resolution for the plots. The development of the ranges w.r.t. the optimisation steps are shown in Figure 6.15.

As depicted in Figure 6.16, the error rate converges to its minimum, while the radius takes on values close to 1. The optimisation parameter λ rapidly converges to its minimum as can be seen in Figure 6.16(c).



(a) Tuning of SVDD parameter C.



(b) Tuning of SVDD parameter σ.

Figure 6.15: Tuning of SVDD parameters using grid search on the "banana", "2 cluster", "Iris", and "thyroid" data set.

Figure 6.17 reveals interesting correlations between C, σ, the radius, the error rate, and the number of support vectors. The influence of the parameters on the radius becomes obvious. As shown by Figure 6.17(a) the radius is anti-proportional to σ. For small values of C, smaller values for the radius are observed. For higher values of C, more training instances are enclosed by the hypersphere, yielding a bigger radius.

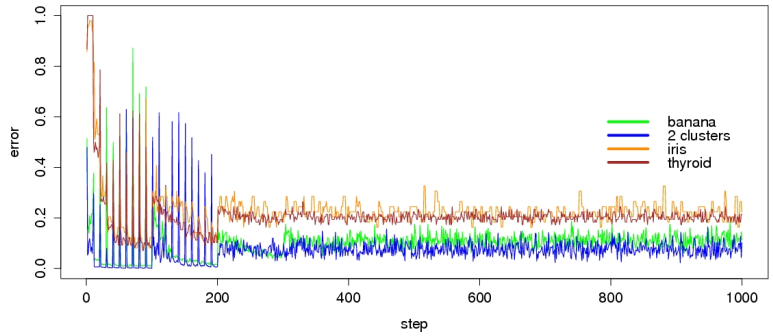
As C increases, the error rate e_{ω_n} decreases, which is easily understood. The parameter C regularises the fraction of outliers in the training set. So, for high values of C, the decision boundary tries to enclose all instances, while for low values of C a higher number of instances is left outside. This leads to a higher number of misclassified instances in the validation set and hence to a higher error rate. Leaving out instances

leads to a flexible boundary, requiring more support vectors. This explains why the number of support vectors decreases as C increases as shown in Figure 6.17(d).

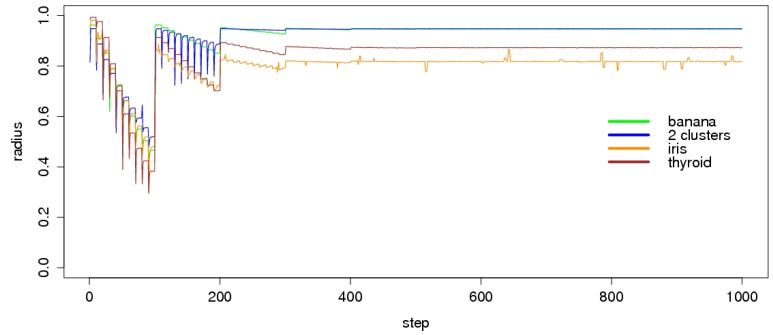
Conclusively, the influence of the SVDD parameters C and σ on the solution are summarised as follows:

$$C \uparrow \quad \Rightarrow \quad \begin{cases} R \uparrow \\ error, SVs, \lambda \downarrow \end{cases} \tag{6.55}$$

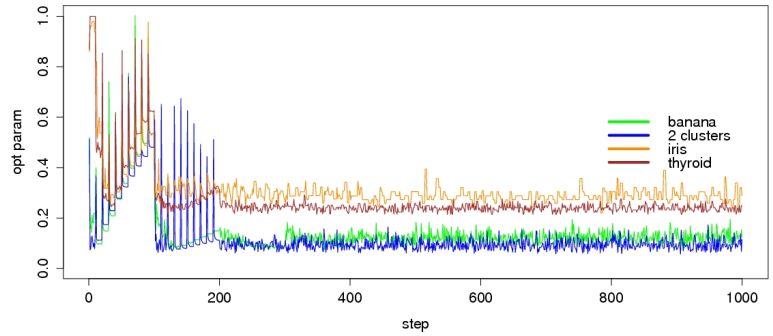
$$\sigma \uparrow \quad \Rightarrow \quad \begin{cases} \lambda \uparrow \\ error, SVs, R \downarrow \end{cases} \tag{6.56}$$



(a) Error rate e_{ω_n} w.r.t. the optimisation step.

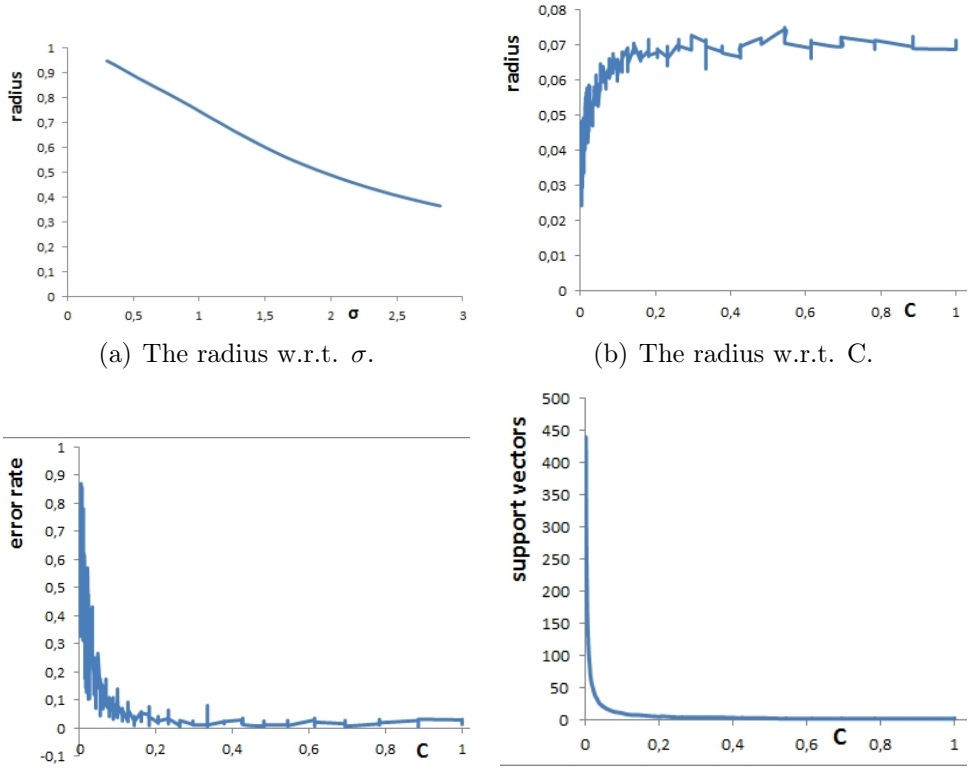


(b) Radius w.r.t. the optimisation step.



(c) Optimisation parameter λ w.r.t. the optimisation step.

Figure 6.16: Error rate e_{ω_n}, radius R, and optimisation parameter λ.



(a) The radius w.r.t. σ.

(b) The radius w.r.t. C.

(c) The error on the normal class e_{ω_n} w.r.t. C.

(d) The number of support vectors w.r.t. C.

Figure 6.17: Correlation between σ and the radius. C influences the radius, the number of support vectors and e_{ω_n}.

6.3.7 Discussion

In the lack of a test set with both classes ω_n and ω_a, a found decision boundary could be evaluated by visualising the training data set with support vectors, as can be seen in Figure 6.14. However, this is not possible for higher-dimensional data sets, since more than two features make it hard to evaluate the visualisation.

It is essential for the usability of the entire approach presented in this Thesis to have an autonomous way to determine the parameters. Users are interested in reported abnormal subsequences in the recordings for further analysis, but should not have to get involved with classification theory and the inner workings of the classifier. The proposed parameter tuning approach offers a major benefit, a valid solution was found that entirely works on the training data set, without any parameterisation.

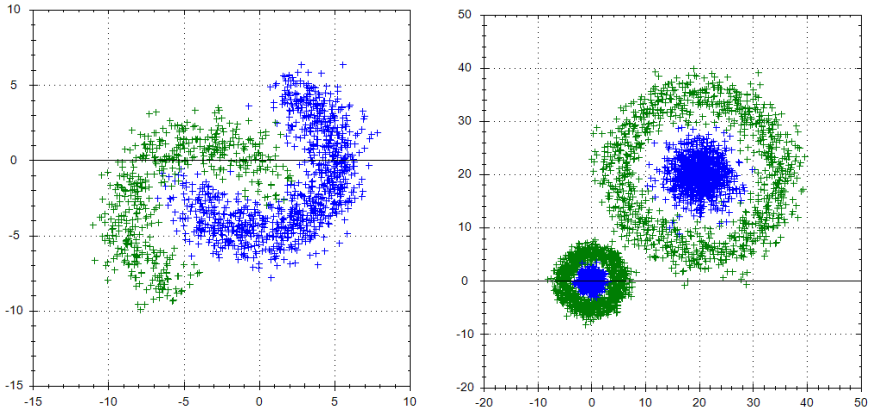
Summarising, the proposed tuning approach finds the optimal parameter set based on the error rate and the radius solely on the training set and without the need for the user to adjust parameters.

6.4 Experimental results on artificial and public domain data

In this section the three discussed one-class classifiers k-NN, LOF, and SVDD are compared based on experimental results on artificial or public domain data sets ranging from 2 up to 60 features. The characteristics of the data sets are described in the first section. Subsequently experiments are conducted with thresholded k-NN as discussed in Section 6.1, followed by experiments with LOF as discussed in Section 6.2. Finally SVDD with the proposed autonomous parameter tuning approach is tested on the same data sets.

6.4.1 Description of the data sets

The two-dimensional "banana" data set, shown in Figure 6.18(a), was created with PRTools (PRTools, 2012) and consists of two classes that are linearly inseparable. The "2 clusters" data set is an own artificial two-dimensional data set, where the normal class comprises two clusters with different densities following a Gaussian distribution and the abnormal class is a ring around each cluster as shown in Figure 6.18(b).



(a) "banana" data set. (b) "2 clusters" data set.

Figure 6.18: Two artificial data sets used for the experiments: the "banana", and the "2 clusters" data set (blue: normal class, green: abnormal class).

Since widely used, Fisher's four-dimensional "Iris" data set was utilised. The class "versicolour" was taken as the normal class, where the training was conducted on the first 50% of instances.

Further real-world data sets were taken from (KEEL, 2012), where more detailed information can be found. The "wine" data set has 13 features, e.g. alcohol or magnesium, determined by chemical analysis of Italian wines from three different cultivars. The first cultivar was used as the normal class.

For the "vehicle" data set (KEEL, 2012) the task is to classify four types of vehicles based on 18 features extracted from their silhouettes viewed from different angles. The class "van" was used as the normal class.

The "twonorm" data set (KEEL, 2012) is an artificial data set with two classes, following a multivariate normal distribution in 20 dimensions.

The "thyroid" data set (KEEL, 2012) contains 21 features, that are used to determine whether a patient is normal or suffers from either hyperthyroidism or hypothyroidism. This data set is special in a way, that the number of normal instances is small compared to the abnormal instances.

Finally the "sonar" data set from (KEEL, 2012) was used, where rocks are to be distinguished from mines. Each instance consists of the energy within 60 frequency bands, i.e. the data set has 60 features. The class "rock" was used as the normal class. This data set has many features but only very few training instances.

A general rule for machine learning algorithms is, the more features a data set contains, the bigger the size of the training set should be. The number of features w.r.t. the size of the training set for all data sets is shown in Figure 6.19. From the plot it can be expected that the "banana" and "2 clusters" data set should yield good classification results, while the "sonar" data set is likely to yield weak results. In addition to the number of features and the size of the training set, the accuracy depends on the separability of the classes for each specific data set.

For all data sets, the features were individually normalised to a value range of $[-1; +1]$ prior to classification. The normalisation factors were determined solely from the training set and used to normalise the training and the test set. This way the test set remains a real *blind* test set.

Since solely the classifiers were to be compared, no further steps to improve the classification results were taken. In applications, feature selection and feature reduction like principal component analysis (PCA) can be conducted.

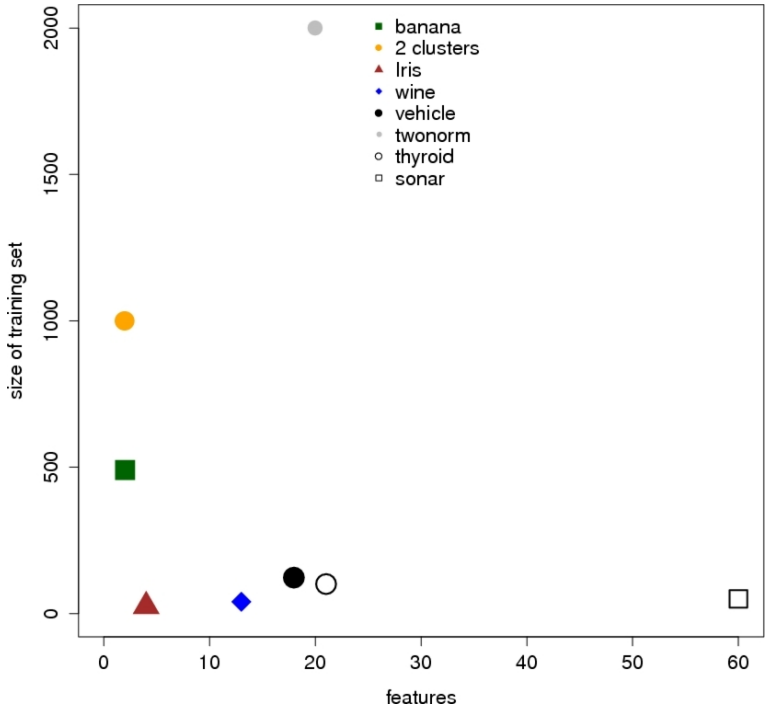


Figure 6.19: Number of features w.r.t. the size of the training set for the 8 data sets.

6.4.2 Results for k-NN, LOF, and SVDD

The results for k-NN with k=1 are given in Table 6.2 and for k=5 are given in Table 6.3, where $\|\mathcal{F}\|$ is the number of features, $\|\mathcal{A}\|$ refers to the number of instances in the training set, and $\|\mathcal{B}\|$ to the number of instances in the test set respectively. The column "threshold" holds the classification threshold determined from the training set. TPR is the true positive rate, i.e. the fraction of normal instances correctly classified as normal, and TNR is the true negative rate, i.e. the fraction of correctly detected anomalies. The last column holds the precision on the abnormal class $\frac{TN}{TN+FN}$.

The k-NN classifier was used with k=1 and k=5, because k=1 is very sensitive to individual outliers and k=5 is less sensitive. As can be seen from Table 6.2 and Table 6.3, the normal class is overestimated. The true positive rate (TPR) is very high, while a high fraction of the data from the abnormal class was not detected as shown by the column holding the true negative rate (TNR). The high true positive rate results in the high percentages for the precision.

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	threshold	TPR	TNR	prec. ω_a
banana	2	490	489/510	0.251	100.0%	72.4%	100.0%
2 clusters	2	1000	1000/2000	0.223	100.0%	51.2%	100.0%
Iris	4	25	25/100	0.687	100.0%	94.0%	100.0%
wine	13	40	19/119	1.751	94.7%	97.5%	99.1%
vehicle	18	122	77/269	1.674	98.7%	58.0%	99.4%
twonorm	20	2000	1703/3697	1.426	99.6%	19.6%	99.2%
thyroid	21	100	66/7034	2.137	98.5%	60.9%	100.0%
sonar	60	50	47/111	4.562	95.7%	18.9%	91.3%

Table 6.2: Results for one-class 1-NN on 8 artificial or public domain data sets ranging from 2 up to 60 features. The fraction of detected anomalies was above 70% for just three of the data sets as shown in the column "TNR".

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	threshold	TPR	TNR	prec. ω_a
banana	2	490	489/510	0.302	100.0%	78.0%	100.0%
2 clusters	2	1000	1000/2000	0.277	100.0%	48.2%	100.0%
Iris	4	25	25/100	1.139	100.0%	93.0%	100.0%
wine	13	40	19/119	2.056	94.7%	98.3%	99.2%
vehicle	18	122	77/269	2.751	100.0%	43.9%	100.0%
twonorm	20	2000	1703/3697	1.545	99.8%	15.1%	99.3%
thyroid	21	100	66/7034	2.448	98.5%	56.5%	100.0%
sonar	60	50	47/111	5.238	100.0%	18.0%	100.0%

Table 6.3: Results on the 8 selected data sets for one-class 5-NN. The results do not significantly differ from 1-NN.

For k=1, the fraction of detected anomalies was above 70% for just three of the data sets. In those data sets, the classes are far apart and can be easily separated. The "sonar" data set yields the weakest results. This data set has a high number of features and a very small size of the training set. This kind of constellation is problematic for classification problems in general, but poses even greater problems for one-class classifiers. The results for k=5 do not significantly differ from the results for k=1.

Analogous to k-NN, LOF was used with k=1 and k=5. Some of the results are very poor as shown in the column holding the true negative rate in Table 6.4 and Table 6.5. In general, LOF is assumed to be superior to k-NN for outlier detection, but the naive way of determining the threshold from the training set appears to be not suitable for LOF.

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	threshold	TPR	TNR	prec. ω_a
banana	2	490	489/510	26.66	99.8%	0.0%	0.0%
2 clusters	2	1000	1000/2000	18.58	100.0%	1.4%	100.0%
Iris	4	25	25/100	1.65	100.0%	84.0%	100.0%
wine	13	40	19/119	1.64	100.0%	89.9%	100.0%
vehicle	18	122	77/269	8.76	100.0%	17.5%	100.0%
twonorm	20	2000	1703/3697	1.86	99.9%	2.4%	98.9%
thyroid	21	100	66/7034	4.37	95.5%	36.2%	99.9%
sonar	60	50	47/111	1.81	95.7%	0.0%	0.0%

Table 6.4: Results for LOF with k=1. The number of detected anomalies is acceptable for two data sets, but very weak on the remaining ones.

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	threshold	TPR	TNR	prec. ω_a
banana	2	490	489/510	3.517	100.0%	36.1%	100.0%
2 clusters	2	1000	1000/2000	2.611	99.5%	91.5%	99.7%
Iris	4	25	25/100	1.323	100.0%	94.0%	100.0%
wine	13	40	19/119	1.295	100.0%	98.3%	100.0%
vehicle	18	122	77/269	2.224	100.0%	50.9%	100.0%
twonorm	20	2000	1703/3697	1.480	99.9%	12.6%	99.6%
thyroid	21	100	66/7034	3.457	97.0%	25.2%	99.9%
sonar	60	50	47/111	1.284	91.5%	17.1%	82.6%

Table 6.5: Results for LOF with k=5. The classification results have improved compared to LOF with k=1 for almost all data sets, but are still very weak for 5 of the 8 data sets.

For k=1, the number of detected anomalies is acceptable for two data sets, but is very weak on the remaining data sets. With k=5, most of the classification results have improved compared to LOF with k=1, but they are still very poor for 4 of the 8 data sets. The reason is the sensitivity to outliers in the training set with k=1. That explains for example the difference between k=1 and k=5 for the "2 clusters" data set.

In the final experiment, SVDD with the proposed autonomous parameter tuning approach was evaluated with the 8 data sets. The experiments were run with 10 parameter candidates, k-fold with k=10, and 10 iterations.

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	SVs	e_{ω_n}	R	TPR	TNR	prec. ω_a
banana	2	490	489/510	23	0.037	0.939	95.1%	96.3%	95.3%
2 clusters	2	1000	1000/2000	36	0.033	0.939	92.7%	99.2%	96.5%
Iris	4	25	25/100	7	0.150	0.777	92.0%	96.0%	98.0%
wine	13	40	19/119	11	0.250	0.837	73.7%	100.0%	96.0%
vehicle	18	122	77/269	20	0.150	0.884	87.0%	82.9%	95.7%
twonorm	20	2000	1703/3697	92	0.050	0.960	94.4%	69.0%	96.4%
thyroid	21	100	66/7034	17	0.160	0.857	71.2%	96.6%	99.7%
sonar	60	50	47/111	12	0.220	0.748	51.1%	38.7%	65.2%

Table 6.6: Results on the 8 selected data sets with SVDD and the proposed autonomous parameter tuning approach. The true negative rate is very good for the majority of the data sets.

Table 6.6 shows the classification results on the described data sets, where SV is the number of determined support vectors, e_{ω_n} the error on the normal class, and R the hypersphere's radius.

As can be seen in Table 6.6, the true negative rate with SVDD is very good for the majority of the data sets, it is above 70% for six of the data sets. As expected, the "sonar" data set yields poor results because with a high number of features and a low number of training instances it is not suitable for classification tasks in general, and particularly poses problems to one-class classifiers. The precision is high, i.e. most of the reported anomalies are abnormal. Additionally, as shown by the column "SVs" the decision functions are expressed by a small fraction of the training instances, e.g. in the case of the "2 clusters" data set, 964 of the 1000 training instances are discarded.

6.5 Experimental results on real data

In this section, the three classifiers are evaluated on own recordings from the test rig with a DC motor and an attached wheel (see Section 3.3.2). Anomalies were manually injected by altering the motor's load simulating e.g. wear-out.

The recordings are time series data, but for these experiments the data points are treated as independent. Ignoring knowledge about dependencies between consecutive

data set	k	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	threshold	TPR	TNR	prec. ω_a
DC motor (IADIS)	1	3	5244	17639/895	0.250	99.9%	22.8%	97.1%
DC motor (IADIS)	5	3	5244	17639/895	0.250	99.9%	22.8%	97.1%
DC motor (position)	1	7	10327	23949/397	0.661	100%	33.2%	96.4%
DC motor (position)	5	7	10327	23949/397	0.709	99.9%	33.2%	91.7%

Table 6.7: Results with k-NN on real data sets from the test rig.

data set	k	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	threshold	TPR	TNR	prec. ω_a
DC motor (IADIS)	1	3	5244	17639/895	19.58	99.9%	0.2%	50.0%
DC motor (IADIS)	5	3	5244	17639/895	3.158	99.9%	27.9%	96.9%
DC motor (position)	1	7	10327	23949/397	36.3	100%	0.02%	100%
DC motor (position)	5	7	10327	23949/397	7.89	100%	0.76%	100%

Table 6.8: Results with LOF on real data sets from the test rig.

data points generally yields weaker results. This issue will be addressed in the next chapter.

The results for k-NN are given in Table 6.7. The percentage of detected abnormal data points is quite low for k=1 and k=5, as shown by the column "TNR". As for the previous experiments, the reason is that the way of determining the threshold from the training data leads to an overestimation of the normal class.

The results for LOF are very poor, which is assumed to be caused by the proposed way of determining the threshold. In particular, with k=1 the results are useless as shown in Table 6.8. The reason is that with k=1 one outlier in the training set determines the threshold.

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$: ω_n/ω_a	SVs	e_{ω_n}	R	TPR	TNR	prec. ω_a
DC motor (IADIS)	3	5244	17639/895	96	0.018	0.980	97.6%	75.2%	61.0%
DC motor (position)	7	10327	23949/397	74	0.008	0.973	97.1 %	70.5 %	28.9 %

Table 6.9: Results with SVDD on real data sets from the test rig.

As for the artificial and public domain data sets, SVDD with autonomous parameter tuning outperforms the one-class adaptations of k-NN and LOF as can be seen in Table 6.9 in the column holding the true negative rate.

6.6 Evaluation

Based on the experimental results, the three classifiers are compared. Figure 6.20 shows the percentage of correctly detected anomalies, i.e. the true negative rate, for the tested data sets. The results reveal that in terms of the fraction of correctly detected anomalies (TNR) SVDD outperforms k-NN and LOF on all data sets.

On the real data sets "DC IADIS" and "DC pos", the results with SVDD were significantly better as shown by the true negative rates in Table 6.9. As indicated by the high percentage rates for TNR, the vast majority of anomalies were detected. In addition the precision on the abnormal class is high, i.e. a high fraction of the reported anomalies is indeed abnormal. The reason is that SVDD encloses the normal data more tightly by describing the decision boundary using the outmost instances as support vectors.

In Table 6.10 the three classifiers are evaluated based on the key requirements identified in Section 5.3 and the classification accuracy. As shown, there is strong evidence, that SVDD is the best classifier for the problem at hand.

Another strong argument for SVDD is that the classification process is very fast compared to k-NN and LOF. Classification is done by solving one equation for the test instance. On the other hand, classification with k-NN and LOF requires visiting each

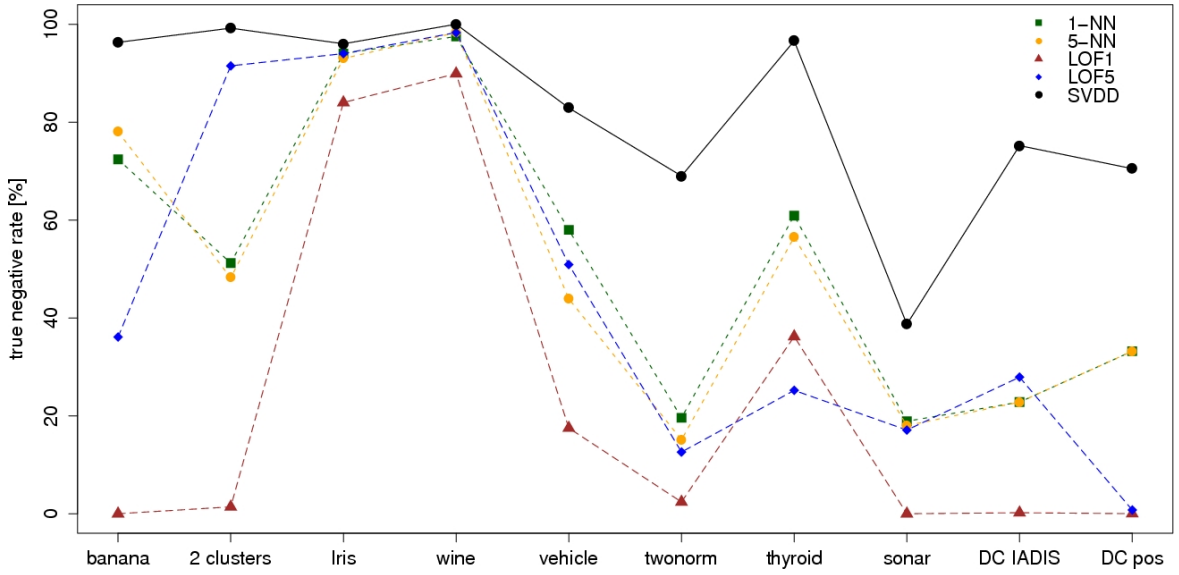


Figure 6.20: True negative rates for the classifiers k-NN, LOF and SVDD on artificial, public domain and real data sets (100% would be optimal).

instance in the training data set. An additional benefit for using SVDD is the very compact way of storing the knowledge base, as indicated by the low number of support vectors.

requirement	one-class k-NN	one-class LOF	SVDD
one-class classifier	yes	yes	yes
robust against outliers	partly	partly	yes
anomaly score	deducible	deducible	deducible
time span	yes	yes	yes
extensible with anomalies	no	no	yes
free of expert-parameters	no	no	yes
sufficient accuracy	no	no	yes

Table 6.10: Evaluation of the three classifiers based on the key requirements from Section 5.3 and the classification accuracy.

6.7 Conclusion

The expectation is that with a more sophisticated way of determining the threshold, the techniques k-NN and LOF could be enhanced to function as a one-class classifier satisfactorily. However, the considered adaptations are not useful.

The discussed benefits make SVDD with the proposed parameter tuning approach most suitable for use in an anomaly detection system. Consequently the decision is to use SVDD for anomaly detection in this Thesis.

In terms of the classification accuracy, by only considering the number of reported feature vectors the results are not fully expressive for the problem to identify abnormal subsequences. For example, five consecutive data points falsely reported as abnormal require the expert to conduct one investigation. Five falsely reported data points scattered in the data set require the expert to investigate five locations in the data set. This problem is addressed in the next chapter.

Chapter 7

Enhancing SVDD to multivariate time series

Having identified SVDD as the most suitable classifier for anomaly detection in the previous chapter, SVDD is enhanced to work on multivariate time series in this chapter. The outcome is a classifier that is directly applicable to test drive data, without complex user parameterisation.

SVDD was identified as the best classifier for anomaly detection, when learning from the normal class ω_n only. However, the classifier works in feature space ignoring the fact that recordings from test drives are time series data. In this chapter a novel approach is proposed, that enhances SVDD to work with multivariate time series. The approach shall be named SVDDsubseq.

7.1 Feature extraction

Classifiers like SVDD are trained with instances in feature space, i.e. feature vectors. Recordings from test drives correspond to multivariate time series data. So from the time series, features need to be extracted. In this work, transforming the multivariate time series to feature vectors is done by transforming the values at each time point T_i to one feature vector F_i. Thereby, an $M \times N$ multivariate time series

$$Y_T = \begin{pmatrix} x_{1,t_1} & \dots & x_{1,t_N} \\ \vdots & \ddots & \vdots \\ x_{M,t_1} & \dots & x_{M,t_N} \end{pmatrix} \tag{7.1}$$

is transformed to N feature vectors of length M

$$F_i = (x_{1,t_i}, x_{2,t_i}, \dots, x_{M,t_i}) \quad \forall i \tag{7.2}$$

7.2 Forming subsequences

The recordings from test drives are noisy. Measuring identical situations, it is likely to observe similar but not identical values. Therefore the expectation is that classifying individual data points yields a high number of false negatives due to the nature of the data, which is confirmed by experiments. An approach is needed, that compensates for small deviations of data points.

The idea is to not classify individual data points, but to incorporate the local neighbourhood of the data points by working on subsequences. From the classified feature vectors, subsequences $Y_{t_j \dots t_{(j+W-1)}}$ are formed using a fixed-width non-overlapping window of length W. The first time point of the j-th subsequence is given by $j * W$ with $j = 0, 1, 2, \dots$:

$$Y_{t_j \ldots t_{(j+W-1)}} = \begin{pmatrix} x_{1,t_j} & \ldots & x_{1,t_{j+W-1}} \\ \vdots & \ddots & \vdots \\ x_{M,t_j} & \ldots & x_{M,t_{j+W-1}} \end{pmatrix} \tag{7.3}$$

Working with feature vectors, the order of the data is ignored. In other words, shuffling the feature vectors (the columns in eq. (7.1)) prior to applying SVDD yields the same results. In contrast, by combining neighbouring values to subsequences, the local order of the data is taken into account.

7.3 Assigning distances to subsequences

In order for subsequences to be classifiable, a distance measure has to be defined. Informally spoken, the distance measure should yield a big distance for a subsequence if many data points lie outside the normal region or if few data points lie far outside the normal region. As a first step, for every feature vector, the Euclidean distance to the center a is calculated by

$$dist_{x_{t_j}} = \|x_{t_j} - a\| \tag{7.4}$$

Based on this distance measure, a distance is assigned to each subsequence. The distance of a subsequence is calculated by summing up the distances of the window's instances.

$$dist_{subseq} = \frac{1}{W} \sum_{j=1}^{W} dist_{x_{t_j}} \tag{7.5}$$

The formation of subsequences after classification of the feature vectors with SVDD is illustrated in Figure 7.1 for a contrived multivariate time series containing two univariate time series.

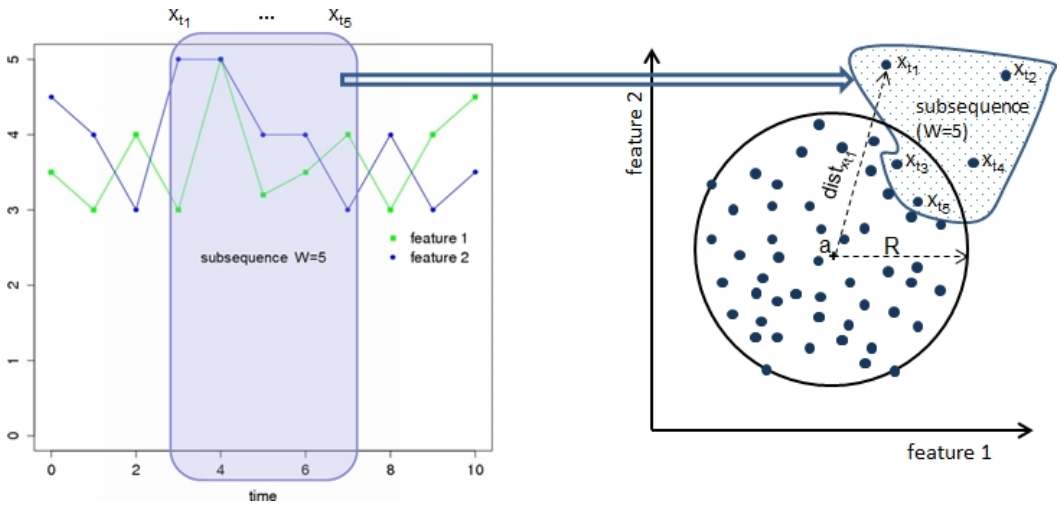


Figure 7.1: A subsequence with window length $W=5$ shown in the original multivariate time series and in feature space.

7.4 Training and test

This section describes the training and test mode of SVDDSUBSEQ. As a first step, SVDD is trained on $\mathcal{A}$ treating data points as independent feature vectors as described in Section 7.1. Following that, for $\mathcal{A}$ the distances to the center are calculcated, transformed to subsequences, and a threshold for the distance $threshold_{subseq}$ is determined.

The procedure during training is as follows:

1. train SVDD with the feature vectors in $\mathcal{A}$
2. calculate the distances $dist_{x_{t_j}}$ of the feature vectors in $\mathcal{A}$, if available an additional tuning set $\mathcal{A}_2$ can be used
3. form subsequences according to eq. (7.3)
4. calculcate $dist_{subseq}$ for all subsequences as given by eq. (7.5)
5. determine a threshold $threshold_{subseq}$ for $dist_{subseq}$

Testing instances from a test set $\mathcal{B}$ works by applying the $threshold_{subseq}$ determined during training:

1. calculate the distances $dist_{x_{t_j}}$ of the feature vectors in test set $\mathcal{B}$
2. form subsequences according to eq. (7.3)
3. calculcate $dist_{subseq}$ for all subsequences (see eq. (7.5))
4. classify subsequences as abnormal if $dist_{subseq} > threshold_{subseq}$

A major advantage of SVDDSUBSEQ is that from the distance values in eq. (7.5), an anomaly score for subsequences can be deduced. It is defined as

$$anomaly\_score_{subseq} = \begin{cases} dist_{subseq} - threshold_{subseq} : & \text{if subsequence is abnormal} \\ 0 : & \text{if subsequence is normal} \end{cases} \tag{7.6}$$

7.5 Determining the threshold

In the previous section the threshold $threshold_{subseq}$ for classifying subsquences based on their distance was introduced. This section discusses a way to determine that threshold.

A first approach to determine the threshold could be to use the maximum distance in $\mathcal{A}$ as the threshold. However, this is highly sensitive to outliers in $\mathcal{A}$ since the threshold would be determined solely by the most distant subsequence.

Equivalently to the introduction of slack variables during training of SVDD, it is proposed to not necessarily include all subsequences in the determination of the threshold, and thereby be robust against outliers. The distances of all subsequences are calculated and those that are considered outliers are not used to determine the threshold.

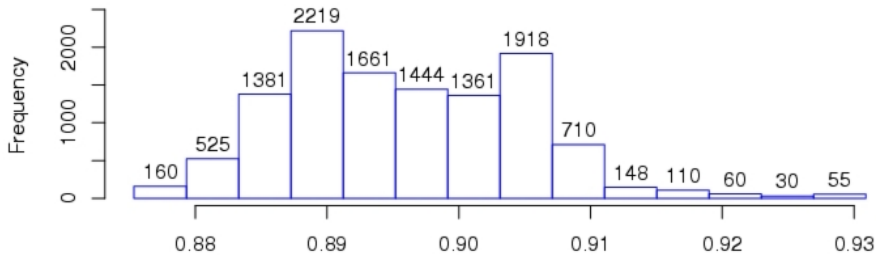


Figure 7.2: Histogram of distances in training set from recordings of test drives.

The undesired outliers in the training set could be identified based on the statistical distribution of the distances by cutting off at the upper tail of the distribution. Experiments have shown that the distribution does not correspond to a normal distribution (see the example in Figure 7.2), so the type of the distribution and its parameters would have to be determined and a threshold would have to be specified.

An approach to determine the threshold without assumptions about the distribution of the distances is desired. It is proposed to use box plots known from statistics (see e.g. (Raykov and Marcoulides, 2012)). For a box plot the first and the third quartile (Q_1 and Q_3) of the data are calculated. The margin between Q_1 and Q_3 is referred to as the inter-quartile range, which holds 50% of the data. Based on the inter-quartile range, the so-called whiskers are calculated by $Q_3 + 1.5(Q_3 - Q_1)$ and $Q_1 - 1.5(Q_3 - Q_1)$. The data outside the whiskers are regarded as outliers. This has been successfully applied on real-world data in (Laurikkala et al., 2000) to identify outliers in medical data.

In this work, outlier distances are the ones that are greater than the upper whisker. Those distances are discarded according to

$$dist_{outlier} > 1.5(Q_3 - Q_1) + Q3 \tag{7.7}$$

and then the maximum of the remaining distances is used as the threshold for classification. Figure 7.3 shows the box plot for the distances shown in Figure 7.2, where

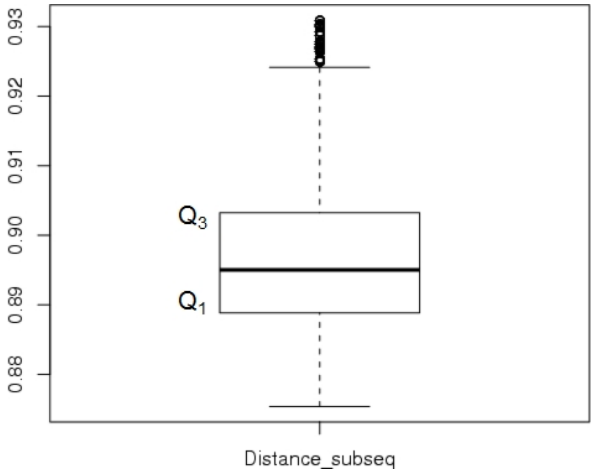


Figure 7.3: Box plot of distances in tuning set.

the distances above the upper horizontal line (whisker) are discarded resulting in a threshold of 0.9214.

7.6 Determining the classification results

In the test set, consecutive fixed-length subsequences with the same label are grouped together as variable-length subsequences, $s_{lab_{\omega_a}}$ for abnormal subsequences and $s_{lab_{\omega_n}}$ for normal ones respectively.

If some or all fixed-length subsequences contained in $s_{lab_{\omega_a}}$ are reported as abnormal, the anomaly is detected, i.e. the system detected one true negative. The classification results are determined as follows, where $s_{class_{\omega_a}}$ is a fixed-length subsequence classified as abnormal and $s_{class_{\omega_n}}$ a subsequence classified as normal. An example is shown in Figure 7.4.

- true negative (TN), i.e. an abnormal subsequence is detected:
  if $s_{lab_{\omega_a}}$ contains at least one $s_{class_{\omega_a}}$
- false positive (FP), i.e. an abnormal subsequence is not detected:
  if $s_{lab_{\omega_a}}$ does not contain a $s_{class_{\omega_a}}$

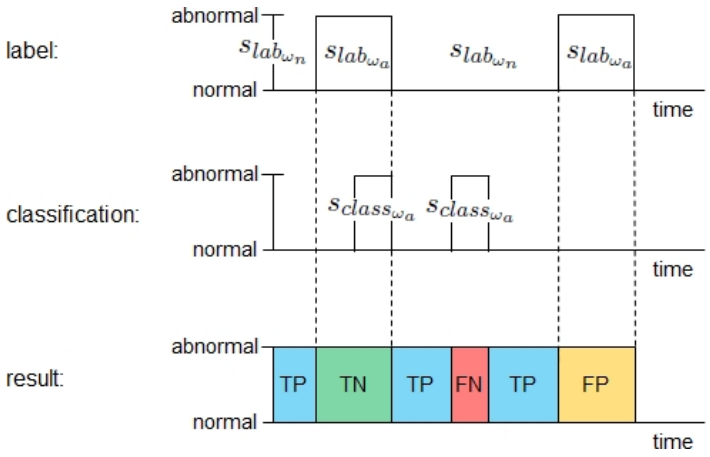


Figure 7.4: Determination of the classification results based on the formed subsequences.

- false negative (FN), i.e. a subsequence is falsely reported as anomaly:
  for each group of consecutive subsequences $s_{class_{\omega_a}}$ contained in a $s_{lab_{\omega_n}}$

- true positive (TP):
  for each group of consecutive subsequences $s_{class_{\omega_n}}$ contained in a $s_{lab_{\omega_n}}$

It will be shown, that the approach is capable to detect anomalies in individual subsequences in a univariate time series and in sets of coinstantaneous subsequences in a multivariate time series, i.e. for a subset of the anomalies defined in Section 3.3. For simplicity the distinction between subsequences and sets of subsequences will not be made in the experiments. A window in a time series will be referred to as "subsequence" regardless of whether it is univariate or multivariate.

7.7 Experimental results on artificial data sets

Having introduced SVDDSUBSEQ, in this section the approach is evaluated using artificial time series data. First, the data sets are described then experiments with different data sets and anomalies are conducted. Finally the results are summarised and conclusions are drawn.

7.7.1 Description of the data set

In order to be able to control the data sets for the initial experiments with SVDDSUBSEQ, time series data was artificially generated. For this purpose, autoregressive moving average (ARMA) models (Shumway and Stoffer, 2006) from the field of time series analysis are used.

An ARMA model is a combination of an autoregressive (AR) and a moving average (MA) model. These three models are briefly reviewed in this section. More information can for example be found in (Shumway and Stoffer, 2006; Kirchgässner and Wolters, 2007).

An autoregressive model yields the current value x_t of a univariate time series w.r.t. its past p values. An AR model is formulated as follows, where α_i are coefficients, w_t denotes white noise, and p is the order.

$$x_t = \alpha_1 x_{t-1} + ... + \alpha_p x_{t-p} + w_t \tag{7.8}$$

With a moving average model, the current value of a univariate time series is modelled as the weighted average of a white noise process w_t. A MA model is given by the following equation, where β_i denotes the coefficients.

$$x_t = \beta_0 w_{t-1} + \beta_1 w_t + \beta_2 w_{t-2} \tag{7.9}$$

ARMA models combine the properties of AR and MA models and are given by

$$x_t = \alpha_1 x_{t-1} + ... + \alpha_p x_{t-p} + w_t + \beta_0 w_{t-1} + ... + \beta_q w_{t-q}, \tag{7.10}$$

An example of a time series generated with an ARMA model with $\alpha_1 = 0.7$ and $\beta_1 = 0.3$ is shown in Figure 7.5.

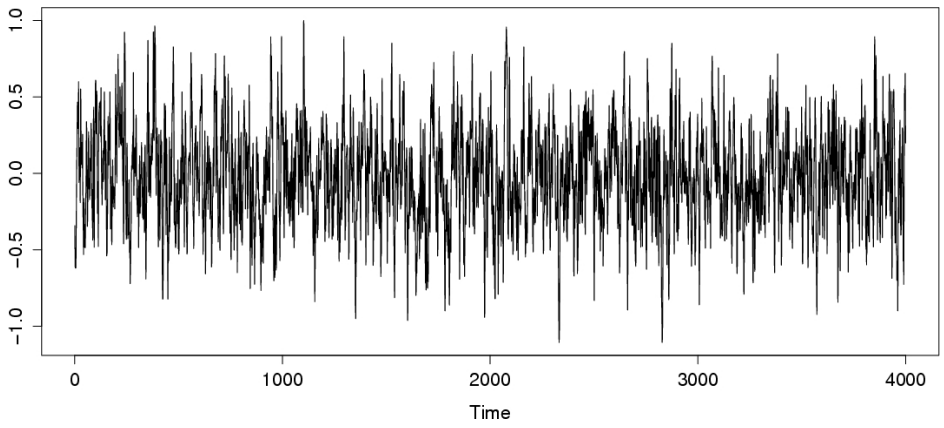


Figure 7.5: Plot of a time series generated by an ARMA model with $\alpha_1 = 0.7$ and $\beta_1 = 0.3$.

The described ARMA models are used for the generation of artificial data sets by generating signals and to add noise to the relations between signals. For all experiments the size of the training set is 5000 feature vectors, the test sets contain 2000 feature vectors. Each test set contains 10 anomalies of different lengths, ranging from 5 to 30 data points. For each experiment three test sets were tested and the results were averaged. A window size W of 10 data points was pre-defined.

7.7.2 Results

In this section, the classification accuracy of SVDDSUBSEQ is evaluated for different kinds of data sets and anomalies. Data sets with a growing number of signals are tested, where the anomalies are present (1) in one signal, (2) in the relation between related signals, and (3) in the relation between two signals with further signals being unrelated. By enclosing the data points in feature space, SVDD implicitly learns the point-wise relations between the signals. As a consequence, the expectation is that the approach works best for closely related signals, while the results are expected to be weaker if unrelated signals are contained in the test set.

7.7.2.1 Anomalies in one of the signals

The first experiment investigates the detection of anomalies that are present in just one of the signals. This type of anomaly was introduced in Section 3.3 as "subsequence anomaly in univariate time series" (type 1). Data sets with related signals were generated utilising the described ARMA models. The training set consists of 5000 data points and the test set of 2000 data points with 10 injected anomalies. The first signal "sig1" is created by an ARMA model. For the further signals, signal n corresponds to signal n-1 superimposed by noise generated by an ARMA model. The anomalies were injected into the second signal. The process of how the data sets were created is shown in Figure 7.6.

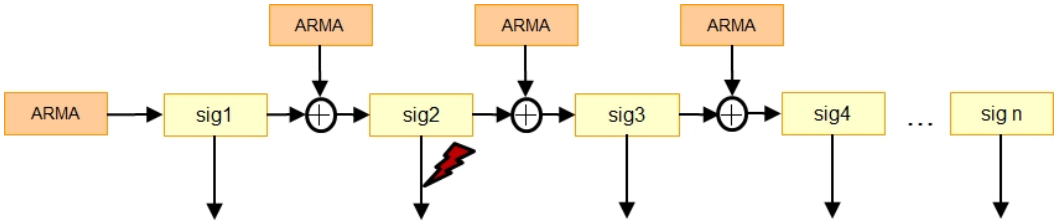


Figure 7.6: Process to create the data set. Signal "sig n" corresponds to "sig n-1" superimposed by noise generated by an ARMA model. The red arrow marks the location where the anomalies are injected.

Figure 7.7 shows one of the multivariate time series with 4 related signals, where the second signal "sig2" contains 10 subsequences that slightly exceed the normal value range learnt from the training set. As can be seen in Figure 7.7, the longer subsequences could be visually detected by an expert investigating the data at this resolution. The smaller subsequences are not obvious though. All anomalies were detected by SVDDSUBSEQ, and 4 false negatives were reported.

The results for 2 ... 20 signals are given in Table 7.1, where $\|\mathcal{F}\|$ is the number of features, i.e. signals, and FN the number of subsequences that were falsely reported as abnormal. The column "avg W" holds the average subsequence length of the detected false negatives and "data points" is the percentage of the falsely reported data points. TN is the number of correctly detected abnormal subsequences (true negatives) and TNR the percentage of detected anomalies. Finally, "precision" is the percentage of true negatives in the result set as introduced in Section 5.2.

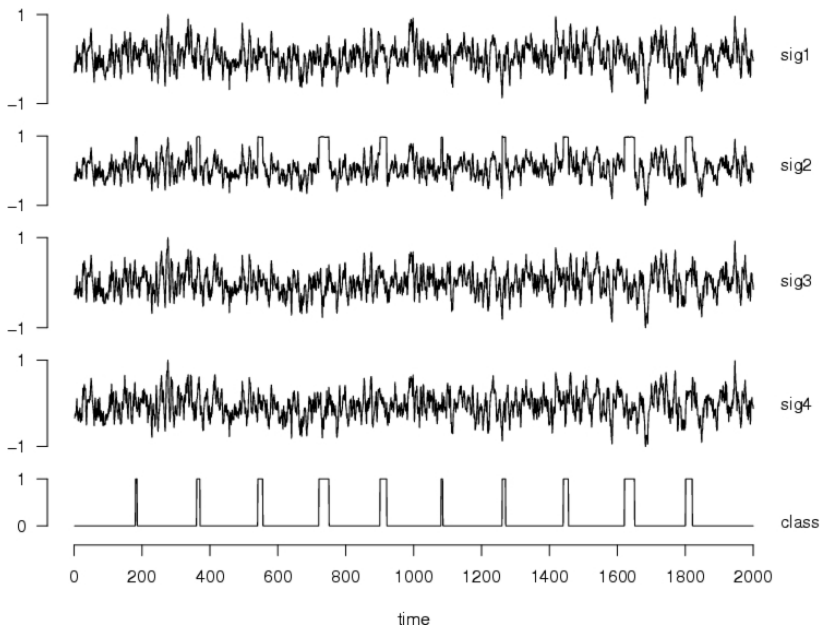


Figure 7.7: A four-dimensional multivariate time series generated using ARMA models. The second signal "sig2" contains 10 anomalies indicated by a value of 1 for "class".

data set	$\|\mathcal{F}\|$	FN	avg W (FN)	data points (FN)	TN	TNR	precision
arma2d	2	4	10	2.0%	10	100%	71.4%
arma4d	4	4	10	2.0%	10	100%	71.4%
arma6d	6	5	10	2.5%	10	100%	66.7%
arma8d	8	4	10	2.0%	10	100%	71.4%
arma10d	10	7	10	3.5%	10	100%	58.9%
arma20d	20	13	11	7.2%	10	100%	43.4%

Table 7.1: Results for 2 ... 20 related signals, where the second signal contains 10 univariate subsequence anomalies.

7.7.2.2 Anomalies in multiple relationships

In the next experiment, anomalies of the type "contextual anomaly in multivariate time series" (type 3, see Section 3.3) are to be detected for data sets with related signals, where the relation is not time-delayed. The first signal "sig1" is created by an ARMA model and further signals correspond to signal n-1 superimposed by ARMA-generated noise. In the test set, 10 anomalies were injected by damping the relationship between two consecutive signals, i.e. the first and the second, the third and the fourth and so forth. The creation of the data set is depicted in Figure 7.8.

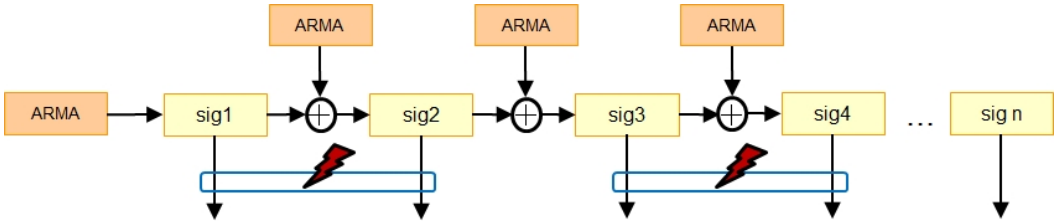


Figure 7.8: Creation of the data sets with related signals and anomalies injected into the relations.

As can be seen from Figure 7.9, it is not possible to visually detect the anomalies. The classifier SVDDSUBSEQ detected 9 of 10 anomalies and 6 false negatives were reported. The full results are given in Table 7.2.

data set	$\|\mathcal{F}\|$	FN	avg W (FN)	data points (FN)	TN	TNR	precision
arma2d	2	5	10	2.5%	7	70%	58.3%
arma4d	4	12	10	6.0%	9	90%	42.9%
arma6d	6	9	11	5.0%	8	80%	47.1%
arma8d	8	7	12	4.2%	9	90%	56.3%
arma10d	10	8	10	4.0%	10	100%	52.9%
arma20d	20	17	11	1.0%	9	90%	34.6%

Table 7.2: Results for 2 . . . 20 related signals, where 10 anomalies were injected into the relations between the first and the second signal, the third and the fourth and so forth.

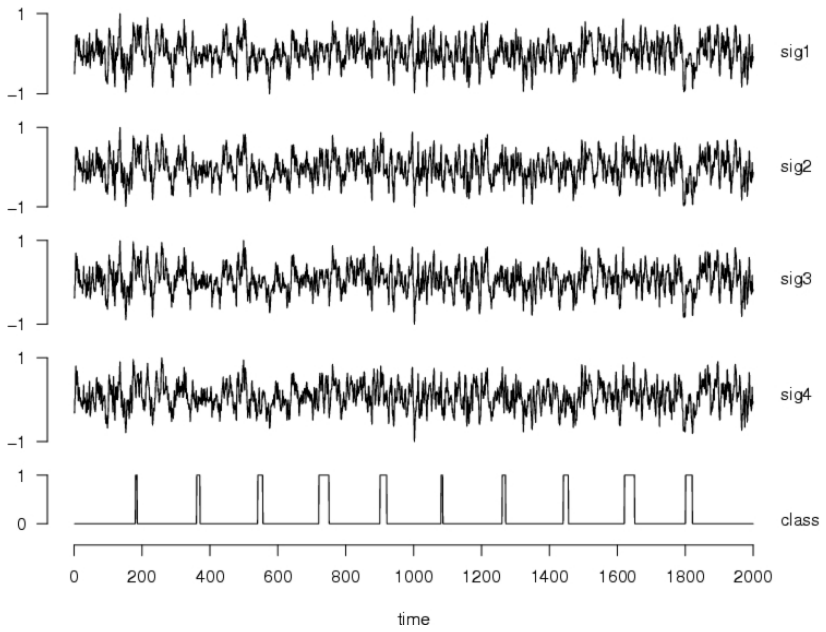


Figure 7.9: Multivariate time series with 4 related signals. 10 anomalies were injected into the relations between "sig1" and "sig2", and between "sig3" and "sig4".

7.7.2.3 Anomalies in one relationship

In the previous example, the anomalies were present in the relationship between several signals. In this experiment just one of the relationships is violated. In the test set, 10 anomalies were injected into the relation between the first and the second signal ("sig1" and "sig2"). The creation of the data set is shown in Figure 7.10. One of the test sets is depicted in Figure 7.11 and the full results are given in Table 7.3. Again, from Figure 7.11 the detection of the anomalies appears to be impossible. Using the proposed classifier, 6 out of 10 anomalies were detected, and 12 subsequences were falsely reported as anomaly.

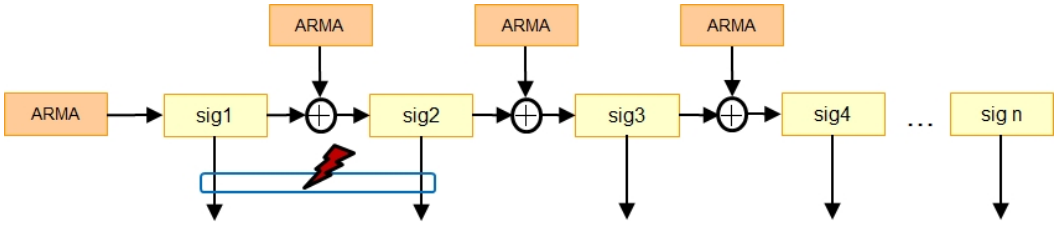


Figure 7.10: Creation of a data set with related signals and injected anomalies.

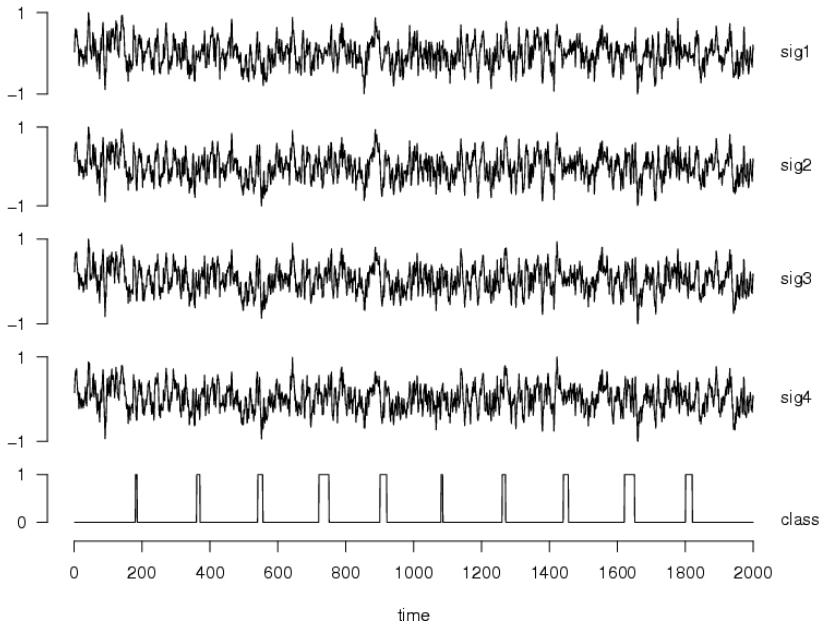


Figure 7.11: Multivariate time series consisting of 4 related signals, where the anomalies were injected into the relation between "sig1" and "sig2".

data set	$\|\mathcal{F}\|$	FN	avg W (FN)	data points (FN)	TN	TNR	precision
arma2d	2	5	10	2.5%	7	70%	58.3%
arma4d	4	8	11	4.4%	5	50%	38.5%
arma6d	6	8	11	4.4%	5	50%	50.0%
arma8d	8	6	11	3.3%	3	30%	33.3%
arma10d	10	8	10	4.0%	3	30%	27.3%
arma20d	20	16	11	8.8%	3	30%	15.8%

Table 7.3: Results for 2 ... 20 related signals, where 10 anomalies were injected into the relation between the first and the second signal.

7.7.2.4 Data sets with unrelated signals

Essentially, the approach learns relationships between signals. In the next experiment the effect of the presence of unrelated signals in the data set is investigated. Data sets with correlations between the first two signals were generated as shown in Figure 7.12. All other signals are random, unrelated signals. 10 anomalies were injected into the relation between the first and the second signal. An example of a test set is given in Figure 7.13, the full results are shown in Table 7.4.

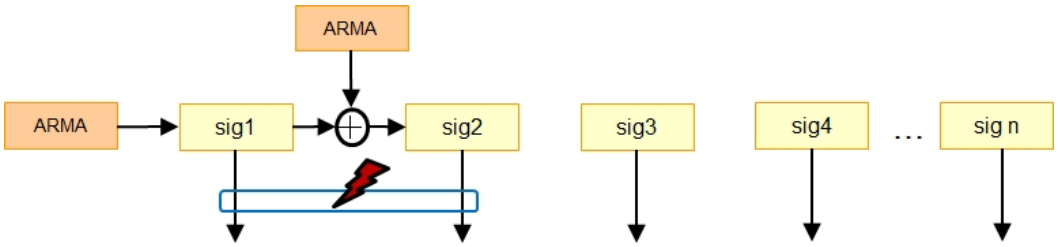


Figure 7.12: Process of creation of a data set with unrelated signals.

data set	$\|\mathcal{F}\|$	FN	avg W (FN)	data points (FN)	TN	TNR	precision
arma2d	2	5	10	2.5%	7	70%	58.3%
arma4d	4	5	10	2.5%	5	50%	58.3%
arma6d	6	17	10	8.5%	2	20%	7.7%
arma8d	8	5	11	2.8%	0	0%	0.0%
arma10d	10	5	10	2.5%	0	0%	25.0%
arma20d	20	13	11	7.2%	2	20%	19.1%

Table 7.4: Results for a growing number of unrelated signals, where the anomalies were injected into the relation between the first and the second signal.

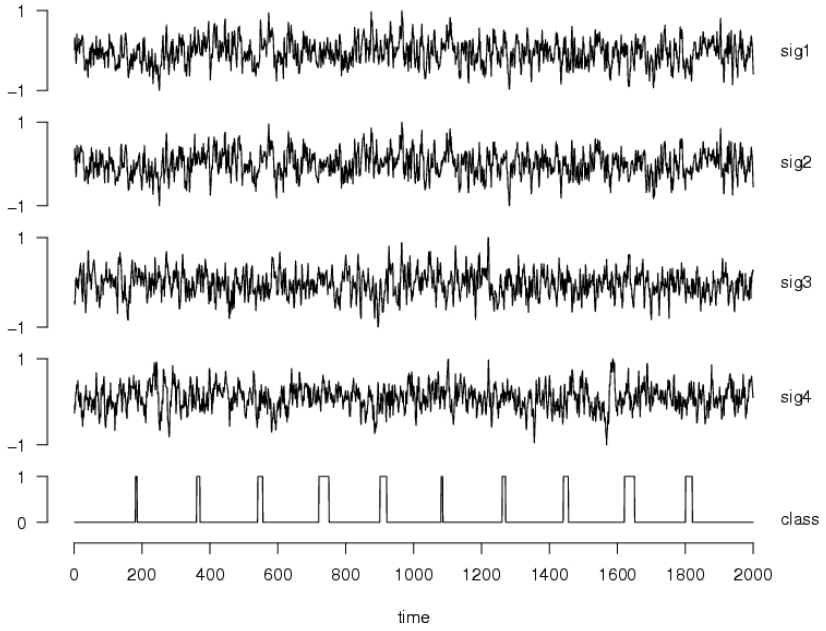


Figure 7.13: Data set with 10 injected anomalies. The anomalies were injected into the relation between "sig1" and "sig2", the signals "sig3" and "sig4" are random, unrelated signals. 6 of the 10 anomalies were detected, 4 were falsely reported as abnormal.

7.7.3 Discussion on results on artificial data

The percentages of correctly detected anomalies (TNR) for the different anomalies and different data sets are shown in Figure 7.14 for one run. The underlying trends are obvious, repeating the experiments with different training and test sets would smooth the curves. It can be seen that all anomalies present in just one signal (type 1) were detected throughout the entire range of features.

The detection rate for the data sets with anomalies in the relationships between two consecutive signals is approximately constant. This is easily understood: by adding more signals, additional signals with anomalies are added. Consequently as the number of signals grows, the number of distinctive features grows as well, allowing the classifier to compensate for the higher number of dimensions.

For the data sets where the anomalies are present in the relation of the first two signals, the TNR decreases to a value of 30% as more signals are added. The reasons are (1) that the additional signals do not add information about the anomaly but the weight of the first two signals decreases as new signals are added, and (2) that an increased number of features requires an increased size of the training set. For the experiments the size of the training set was kept fixed to show the effects of the training set becoming less representative.

A massive decrease of the TNR for a growing number of signals can be observed for the data sets with unrelated signals. This is expected, since it is inherent to all classifiers. Features that do not distinguish the classes should be removed from the training set.

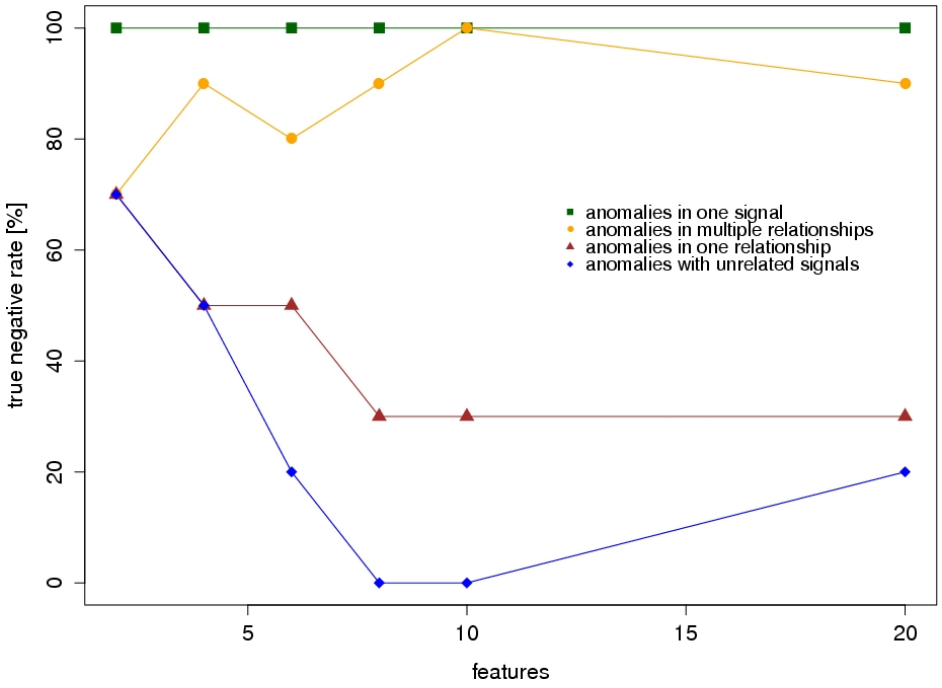


Figure 7.14: Results for different ARMA data sets and anomalies w.r.t. the number of signals.

In general, the number of false negatives increases as the number of signals increases. Since the size of the training set is kept fixed, the training set becomes less representative in more dimensions. For the data set with unrelated signals no increase of the false negatives could be observed in the experiments. The reason is that the random, unrelated signals cover the entire feature space within the signal's min/max values.

From the results a number of conclusions can be drawn:

1. Anomalies, where one of the signals exceeds a value range can be detected.

2. SVDDSUBSEQ performs well on data sets with related signals. In recordings from test drives, groups of signals can be identified, where the signals are correlated. For these kind of data sets, SVDDSUBSEQ will offer the most benefit.

3. In terms of unrelated signals, the approach performs weaker. However, all classifiers are sensitive to irrelevant features. Hence, unrelated signals should be removed prior to training.

7.8 Experimental results on real data

In this section experiments on recordings from the DC motor test rig, that was introduced in Section 3.3.2, are shown and discussed.

7.8.1 Description of the data sets

From the DC motor test rig seven signals can be measured: DC current, actual/set position, actual/set rotation speed, coil temperature, and controller output. By altering the motor's load for a short period of time, anomalies were manually injected. The recordings were separated into a training set with normal data and a test set with normal and abnormal subsequences. A window length W of 10 data points was pre-specified.

7.8.2 Results

The first experiment was conducted with the 3-dimensional "DC motor IADIS" data set. The motor controller was programmed to follow a pre-defined velocity profile.

After inspection using the visual data mining techniques introduced in Chapter 4 the signals DC current, controller output and the motor's acceleration, which was added by deriving the velocity signal, were used. The task is to find abnormal deviations of the motor load. The test set is shown in Figure 7.15, where it can be seen that the injected faults are not obvious. The results are given in the first row in Table 7.5.

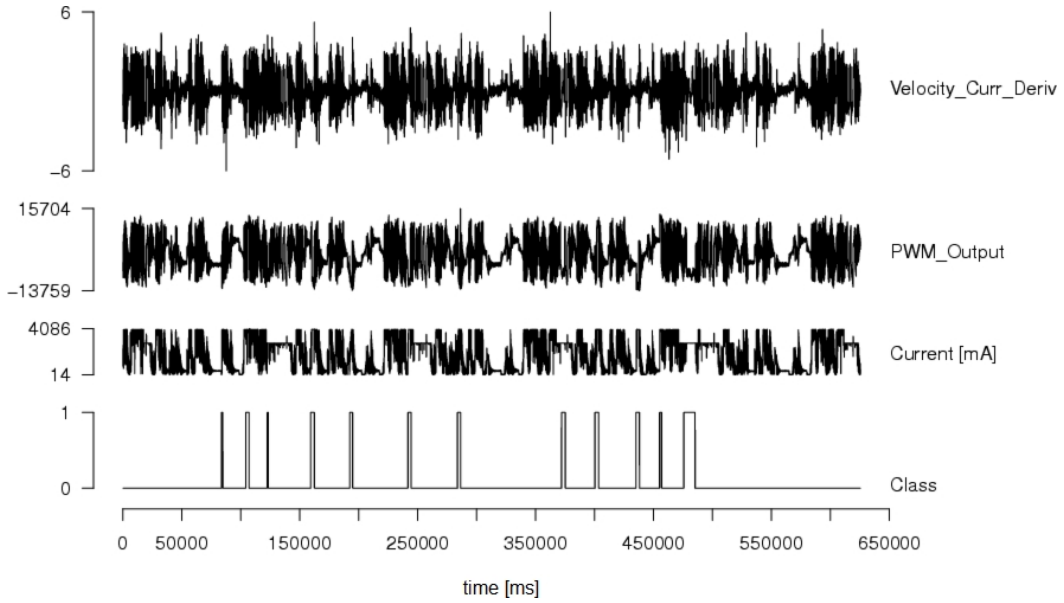


Figure 7.15: Data set "DC motor IADIS" recorded from the DC motor test rig. 12 faults were injected by altering the motor's load.

Two further data sets, with 5 and 7 signals, were recorded while the DC motor was following different profiles of the motor position. The results are shown in the second and third row of Table 7.5.

data set	$\|\mathcal{F}\|$	FN	avg W (FN)	data points (FN)	TN	TNR	precision
DC motor IADIS	3	1	10	0.06%	12	100%	92.3%
DC motor pos1	5	6	10	0.32%	9	100%	60.0%
DC motor pos2	7	22	12	0.54%	20	100%	47.6%

Table 7.5: Results on recordings from the DC motor test rig.

7.8.3 Discussion on results on real data

The results for all data sets from the DC motor were very good. All anomalies were detected and the number of false negatives is acceptably low. The reason is that the motor follows pre-defined profiles, which allows obtaining a representative training set.

In test drives, drivers are involved and the data sets will not be as controlled. For that reason weaker results are expected when applying the approach to recordings from test drives.

7.9 Conclusion

The nature of the time series data from vehicles will cause a high number of normal feature vectors to lie outside the decision boundary, i.e. being falsely classified as abnormal. Reducing the false negative rate could be achieved by influencing the decision boundary to grow bigger than necessary using a cost function. However, this in turn would increase the false positive rate. To overcome this problem, it was proposed to use the knowledge about the local neighbourhood in the data set using the formed subsequences.

The novel approach SVDDSUBSEQ was proposed, enhancing SVDD to work with multivariate time series. The outcome is a classifier that is directly applicable to test drive data, without the need for expert knowledge to configure or tune the classifier. The approach was shown to yield good results for multivariate time series, where the individual time series are related. This makes the approach applicable to test drive data, since in a vehicle groups of signals are related.

Furthermore it was shown that SVDDSUBSEQ is capable to detect two of the three anomaly types introduced in Section 3.3: “subsequence anomaly in univariate time series” (type 1) and “contextual anomaly in multivariate time series” (type 3) when the dependencies are not time-delayed.

Chapter 8

The anomaly detection system

In this chapter, an anomaly detection system is introduced. The detection system uses SVDDsubseq, which was introduced in the previous chapter, accompanied by the enhanced visual data mining techniques introduced in Chapter 4.

As discussed in Chapter 1, the complexity of vehicle electronics will further increase resulting in an increase of the data volume and the complexity of recordings from test drives. This demands advancements in the techniques used for the analysis of the recordings. Only by utilising advanced data analysis techniques, one can make sure that the high effort put in the recording of test drives will continue to pay off in the future. Based on the theoretical background and the experimental results from the previous chapters, an anomaly detection system is proposed in this chapter.

As a prelude to the chapter with experiments, the components of the entire anomaly detection system are described in this chapter. The general steps of a machine learning classification system, that were introduced in Chapter 5, are refined for the proposed anomaly detection system. The steps are shown in Figure 8.1 distinguishing between automatic and user-driven steps.

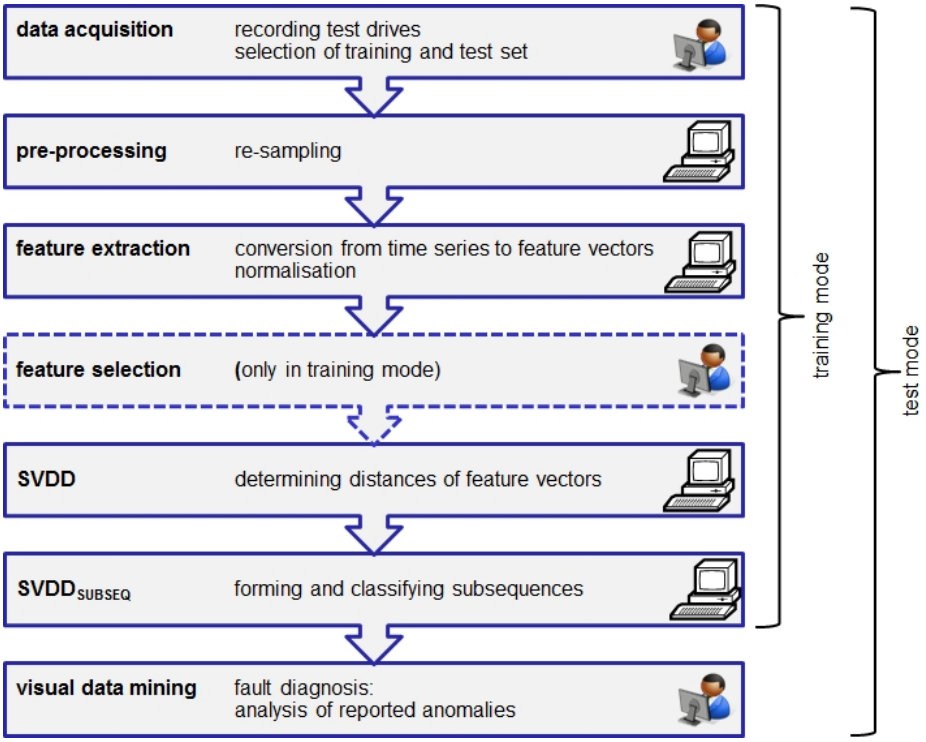


Figure 8.1: Steps of the anomaly detection system, from data acquisition to classification of subsequences and their analysis.

8.1 System overview

Simply stated, the presented anomalies show deviations from behaviour that is viewed as normal. In order to detect anomalies, knowledge about normal behaviour is learnt from a training set of recordings. The following operation modes are proposed for the detection system:

1. training mode
   a) integration of expert knowledge
   b) learning normal behaviour from a training set
2. test mode (anomaly detection)

   a) applying the knowledge base to unseen data

   b) presentation of anomalies

3. feedback mode

   a) manual classification of the results by means of the introduced visual data mining techniques

   b) enhancement of the knowledge base

Figure 8.2 shows the system's training mode, where expert knowledge like the selection of relevant signals is integrated. This can either take place by a domain-expert or in an automatic manner based on input files from the vehicle's development process. Following that, the detection system is trained on a training set of recordings that contain normal data only. The proposed classifier SVDDSUBSEQ is used and the extracted knowledge is stored in a knowledge base.

In test mode, depicted in Figure 8.3, the system detects anomalies. The knowledge base is applied to unseen recordings in order to automatically present the found anomalies together with the points in time they occurred. An anomaly score allows prioritising the results. This way the system points the user to the relevant parts in the recordings.

The third step is referred to as feedback mode. The user analyses the presented anomalies using the visual data mining techniques introduced in Chapter 4. The reported anomalies are manually classified as correct, i.e. fault found, or incorrect. The training set can be enhanced by new recordings or the knowledge base could be enhanced by the expert's classification of the reported anomalies.

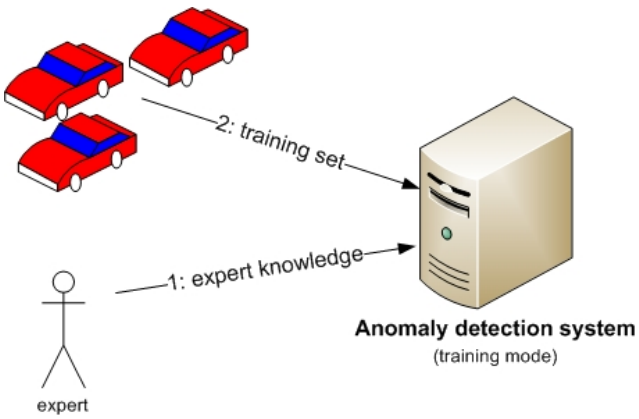


Figure 8.2: Anomaly detection system in training mode. The system is trained on a set of recordings.

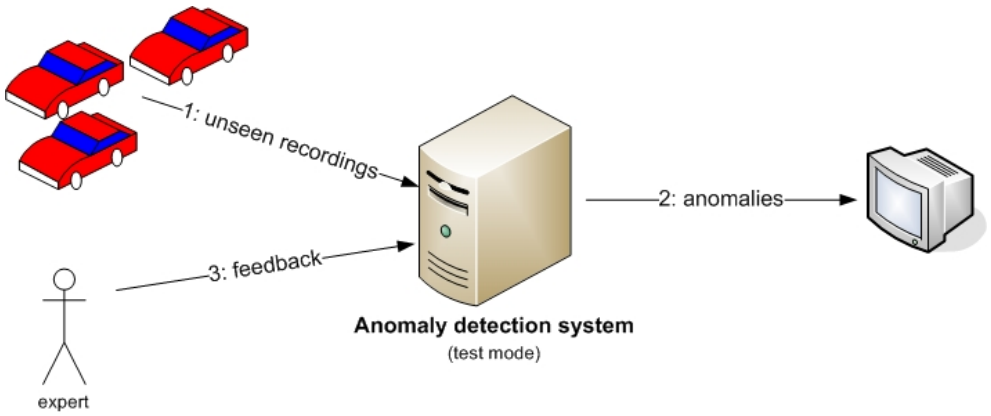


Figure 8.3: Anomaly detection system in test and feedback mode. The knowledge base is applied to unseen recordings and anomalies are reported.

8.2 The implementation

The anomaly detection system was predominantly implemented in the C# programming language. It was implemented as a modular framework that consists of the application, modules, plug-ins, and extension points. The underlying idea is, to have system experts configuring the workflows in the experimental stage. In an industrial project, this could be employees of the tool supplier. During system operation, the pre-configured workflows are run and the results are analysed by data analysts.

A variety of data import readers were implemented in order to be able to read different file formats. Furthermore pre-processing algorithms were implemented that allow to filter signal profiles, or to reduce the number of data points. In addition, various classifiers, visualisation techniques, and reporting facilities were implemented. These individual functions are represented as plug-ins that can be combined to a runnable workflow. This allows for flexible data analysis, which is especially beneficial during the experimental stage. Additionally, the application can interface with the R statistics tool box (R Development Core Team, 2010) and Matlab (MATLAB, 2011), and thereby exchange data with these tools. If new functionality is required, it can be implemented by writing a new plug-in or functions from the R statistics tool box or Matlab can be used.

For the integration of the SVDD functionality, SVM.NET (Johnson, 2009) was used, which is an open-source port of the libSVM (Chang and Lin, 2011) to the C# programming language. libSVM does not contain SVDD, so the SVDD enhancement from (Wang et al., 2010) was migrated to C#, adapted to cover only SVDD with the RBF kernel, and added to the SVM.NET library.

Figure 8.4 shows a configured workflow in the middle pane, where each plug-in is represented by a rectangle, e.g. "SwitchLearningState". An excerpt of the available plug-ins is listed on the left hand side and the plug-in configuration is shown on the right hand side.

In this example workflow, the first plug-in determines whether a training, a test, or a performance run should be conducted. Subsequently for data acquisition a variety of import filters can be selected. The next step is the normalisation of the input data followed by the classifiers SVDD and SVDDSUBSEQ. The last plug-in stores a report.

Workflows can be run on the local computer or transferred to a more powerful server. In the latter case the results are obtained by means of reports. In order to allow for the analysis in batch mode, for example overnight, workflows can be queued on the server system.

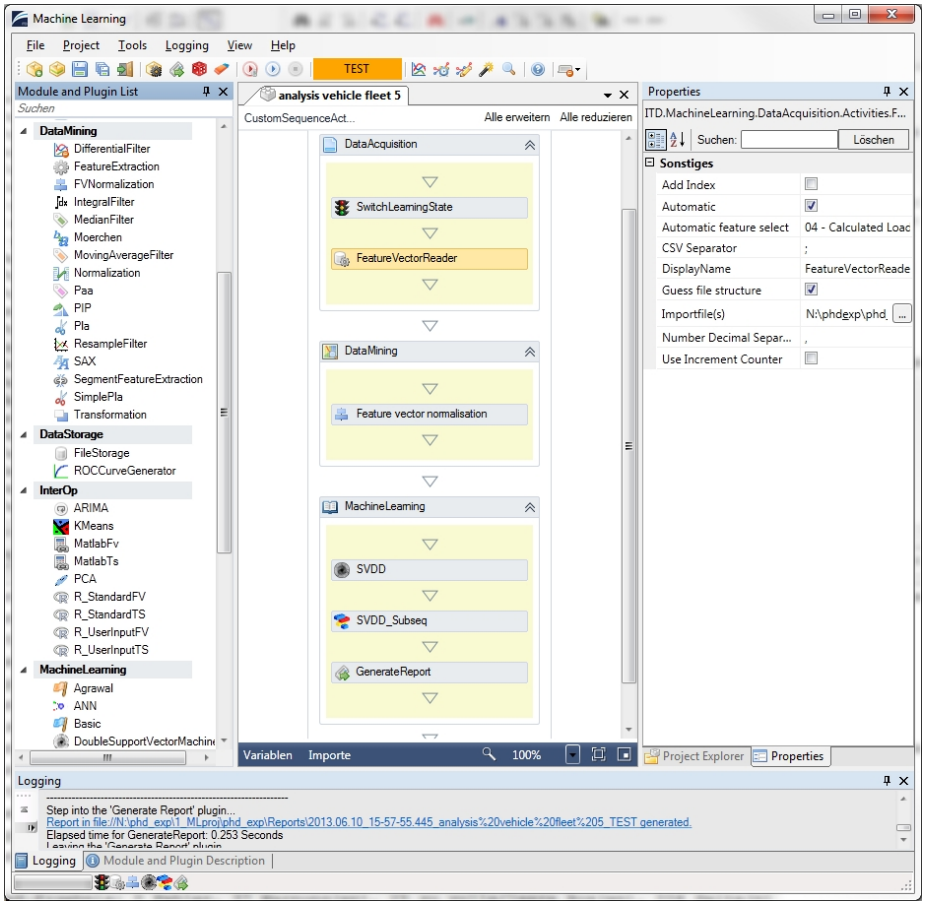
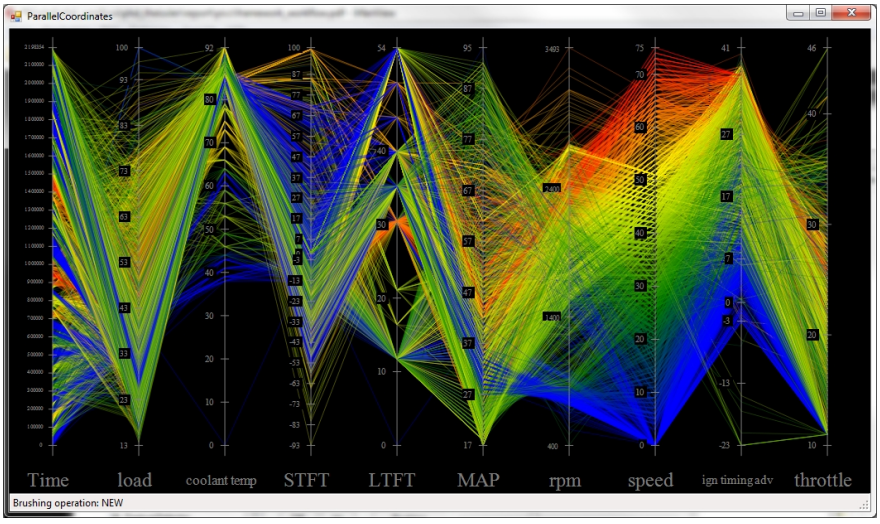


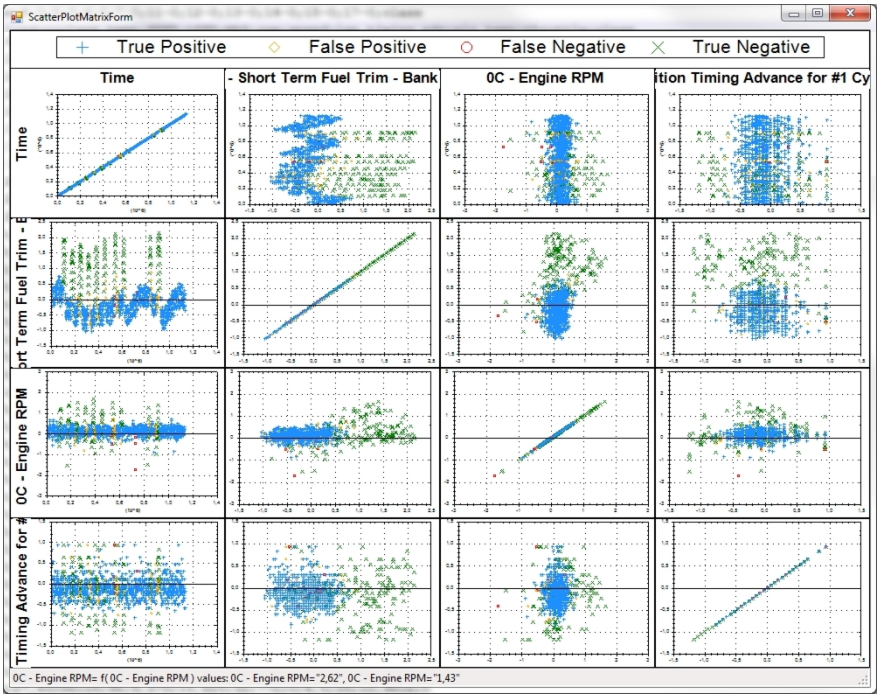
Figure 8.4: The middle pane shows a configured workflow in the anomaly detection system. Available plug-ins are listed on the left hand side and their configuration is shown on the right hand side.

The visualisation techniques introduced in Chapter 4 are integrated as plug-ins. Figure 8.5(a) shows the use of parallel coordinates, which can be used for the analysis of potential training data, e.g. to select the relevant signals, or for the analysis of the reported results.

In Figure 8.5(b), the classification results for a recording from a vehicle in idle mode are shown in a scatter plot matrix. Especially in the experimental stage it is viewed as crucial, to not just analyse the classification results of autonomous classification, but to visualise the results in order to understand potential shortcomings of the classification.



(a) Manual analysis of a test drive using the parallel coordinates plug-in.



(b) The scatter plot matrix plug-in used for the analysis of the classification results for a recording from a vehicle in idle mode.

Figure 8.5: Visual data mining techniques for the analysis of training data or reported anomalies.

8.3 Conclusion

In addition to a theoretical framework this Thesis presents a full implementation of the proposed anomaly detection system. This way, the concept can be proved by validating the full process for anomaly detection in test drive data. It also allows evaluating the system's usability and to gain experience with the processing time the algorithms take for different data sets.

Chapter 9

Experimental results on recordings from vehicles

This chapter shows experimental results on recordings from vehicles. It introduces a way to find a good size for the training set in the absence of abnormal test data and investigates the effect of different driving conditions, different drivers, and different vehicles. The results can be used as a guideline when setting up an anomaly detection system for own vehicle data.

After having shown in Section 7.8 that SVDDsubseq works in a controlled environment, in this chapter the approach is validated on real data sets from vehicles. Over 200 test drives were conducted in different traffic situations ranging from urban traffic to motorways over a time span of one year to capture recordings from different weather conditions. The data acquisition phase started with one vehicle and one driver and was later extended to multiple drivers and four vehicles to become more representative.

Different cars of the type "Renault Twingo" were used as test vehicles due to the availability of the vehicles and the easy accessibility of components in the engine bay.

Figure 9.1: Test vehicle "Renault Twingo" from 2002 with 1149 ccm and 43 kW.

The results are not constrained to this type of vehicle, though. Figure 9.1 shows the test vehicle that is used in the first experiments.

The data during the test drives was recorded using the on-board diagnostics interface according to ISO 15031 (OBD-II or EOBD), that is available in all petrol-driven cars manufactured in 2001 or later, and in 2004 or later for Diesel cars respectively. OBD allows for the approximate reading of 10-15 emission-related signals like the vehicle's speed or the engine rpm in a standardised way. The signals are periodically requested using so-called parameter ids (PIDs). For more information see for example (Denton, 2006). The signals were recorded with a sample rate of one second using 10.4 kBit/s K-Line or 500 kBit/s CAN access (see Section 1.2).

Recordings from test drives from automotive manufacturers are highly confidential and could therefore not be used. Recordings were available to the author but not for publishing purposes. So the author could assure that the data recorded from own test vehicles is comparable, it has a lower sample rate though. The advantage of recording own test drives is that the author had full control over the type of the test drives, the drivers, and fault injection.

The length of the subsequences for SVDDSUBSEQ was set to W=5, i.e. to 5 seconds, for all experiments.

9.1 Injected faults

In order to test the anomaly detection system, faults were injected into the test vehicles to obtain recordings with errors. The aim was to inject faults that manifest themselves as different types of anomalies from Section 3.3. The following four faults were injected:

1. **Fault #1 (unavailable engine temperature):** A loose contact in the wiring of the temperature sensor was simulated by interrupting the connecting wire. This leads to a temperature value of −40 °C. As shown in Figure 9.2 this is out of the value range present for this signal in the training set.

   The fault manifests itself as an anomaly of type 1 ("subsequence anomaly in univariate time series"). Due to the very good results for this anomaly type on the artificial data sets in Section 7.7.2.1, this fault is expected to be reliably detected.

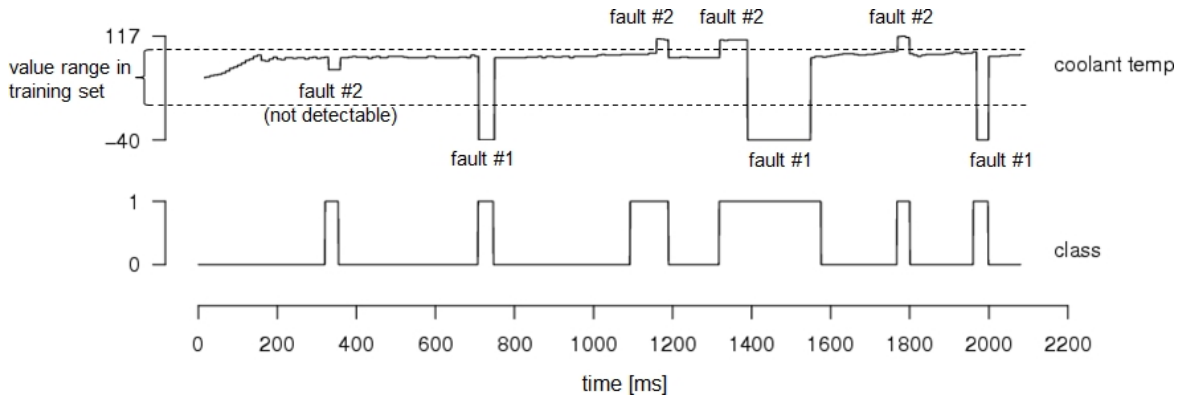


Figure 9.2: A recording of an overland drive with injected faults of type #1 and #2, where a fault is indicated by a value of 1 for "class".

2. **Fault #2 (erroneous engine temperature):** An error in the sensor measuring the engine coolant temperature was simulated. The sensor is a negative temperature coefficient (NTC) thermistor in the test vehicles, i.e. low resistance corresponds to high temperature. In the case of the used vehicle e.g. 7.5 kΩ corresponds to +4 °C and 240 Ω to +90 °C (Etzold, 2011). Using a potentiometer either in a series or a parallel circuit, sensor offsets were simulated.

Injecting a negative sensor offset yields values that are within the valid value range as shown in Figure 9.2. The fault would only be detectable if the relationship to further signals is violated, which is not the case.

Adding a positive offset leads to temperature values of approximately +110 °C. Since such high temperature values are not present in the training set, the fault corresponds to a type 1 anomaly ("subsequence anomaly in univariate time series").

3. **Fault #3 (injection):** Misfiring by an erroneous or coked injector nozzle or a loose contact in wiring was simulated by switching off an injector nozzle for a short period of time, suppressing injection for one cylinder.

   As a countermeasure to the injected fault, the engine control system adapts the injection pulse width, which is observable by a change in the signal "short term fuel trim" (STFT).

   As shown in Figure 9.3, for 4 of the 9 occurrences, the fault manifests itself as a "subsequence anomaly in univariate time series" (type 1), since the values of STFT are greater than the values in the training set.

   The remaining occurrences correspond to a "contextual anomaly in multivariate time series" (type 3) and are only detectable by considering dependent signals.

4. **Fault #4 (ignition):** An erroneous spark plug lead or spark plug was simulated by interrupting one spark plug lead for a short period of time while the vehicle was standing still, simulating a loose connection or a worn spark plug.

   As for fault #3, the signal STFT is adapted by the engine control system. As shown in Figure 9.4 the signal increases but not to values that are out of the normal range.

   The fault should be detectable by considering the relationship to further signals. Such high values of the "STFT" signal while the vehicle speed and the engine revolutions are at low values were not observed under normal operation

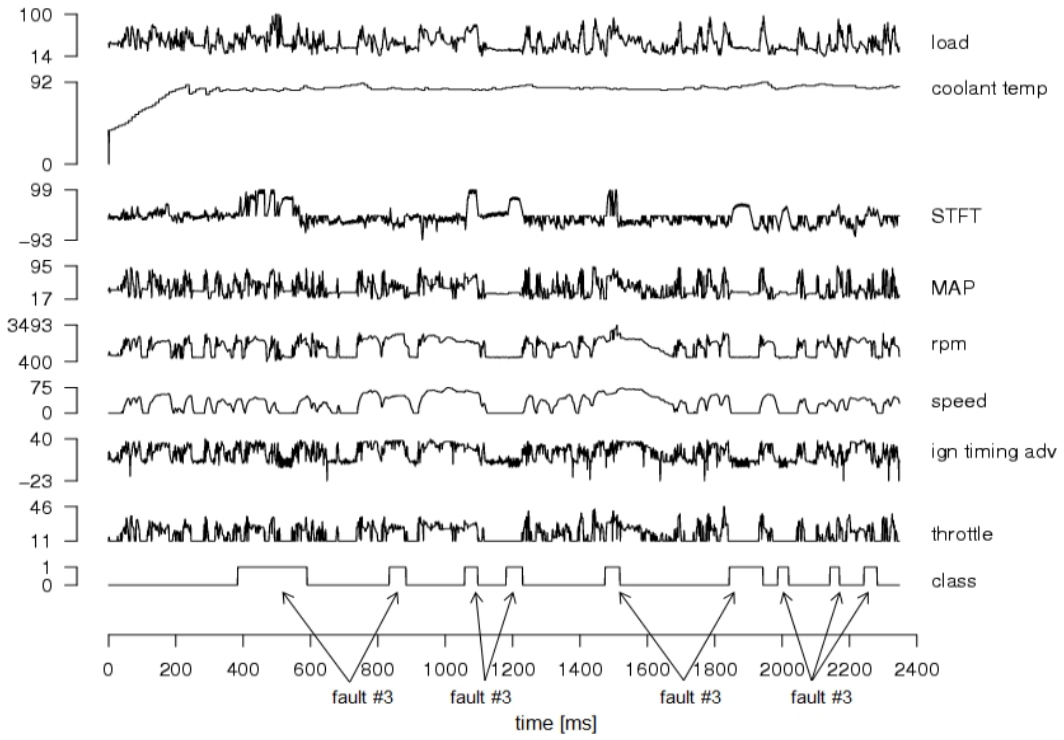


Figure 9.3: Faults of type #3, injected during an overland drive.

mode, which makes this fault an anomaly of type 3, a "contextual anomaly in multivariate time series".

Referring to the locations of potential faults identified in Section 2.4, the injected faults correspond to faults in the cable harness, actuators, or sensors. The mapping of the injected faults to the categorisation of anomalies from Section 3.3 is summarised in Table 9.1.

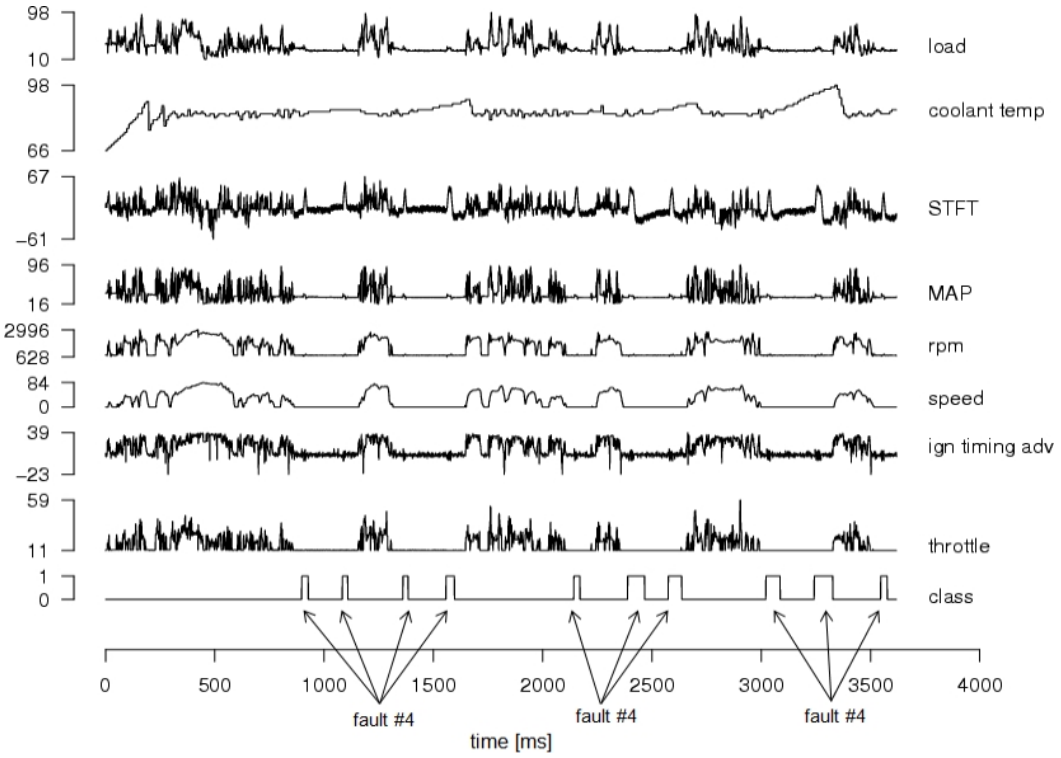


Figure 9.4: An overland drive with injected faults of type #4.

anomaly type	anomaly full name	fault
type 1	subsequence anomaly in univariate time series	fault #1 fault #2 (partly) fault #3 (partly)
type 2	contextual anomaly in univariate time series	not injected because not detectable with the approach
type 3	contextual anomaly in multivariate time series	fault #3 (partly) fault #4

Table 9.1: Anomaly types from Section 3.3 and injected faults.

9.2 Experiments with vehicle in idle mode

The first experiments were conducted on recordings from a vehicle in idle mode in order to investigate the approach without the expected variability of the data in the case of test drives. The signals in Table 9.2 were used for the experiments in idle mode. A "Renault Twingo" manufactured in 2002 was used as the test vehicle.

9.2.1 Experiments on error-free recordings from idle mode

In the absence of abnormal data it is recommended to start by testing with normal data and determining the number of false negatives, i.e. the falsely detected anomalies.

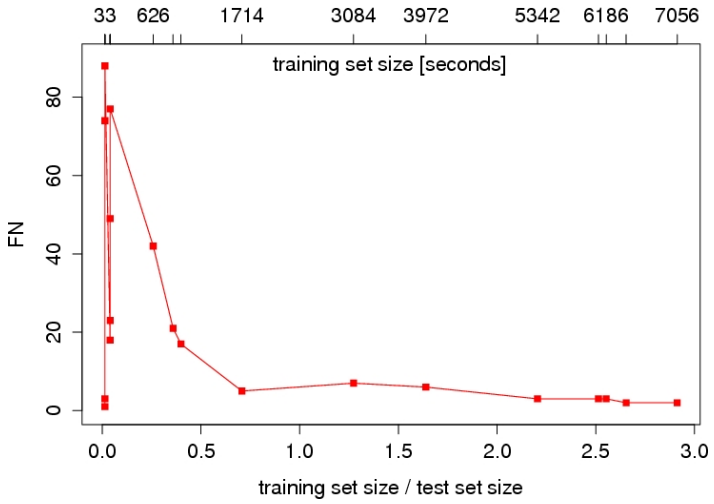


Figure 9.5: Number of false negatives FN w.r.t. the ratio between the size of the training set and a fixed size of the test set (2403 seconds), where the size of the training set is additionally shown by the upper x-axis.

Signal short name	Signal full OBD name	unit	OBD PID
STFT	Short Term Fuel Trim (Bank 1)	%	06 hex
rpm	Engine RPM	rpm	0C hex
ign timing adv	Ignition Timing Advance (Cylinder 1)	°	0E hex

Table 9.2: Signals used for the experiments in idle mode

To investigate the impact of the size of the training set, the size was varied with a fixed test set size of 2403 seconds. The number of false negatives FN w.r.t. the ratio between the size of the training set and the size of the test set $\frac{\|\mathcal{A}\|}{\|\mathcal{B}\|}$ is shown in Figure 9.5. While for very small training sets the number of false negatives acts non-deterministically between very low and very high values, for larger training sets, the training set becomes more representative and the number of false negatives stabilises at low values. This type of experiment can be used as an indicator of how representative the training set is.

From Figure 9.5 the minimal size of a good training set can be deduced. It is the point where the number of false negatives starts to stabilise at low values: the training set size of 1714 seconds. The size is entirely data-dependent, and it is recommended to conduct this experiment as a first step for available data.

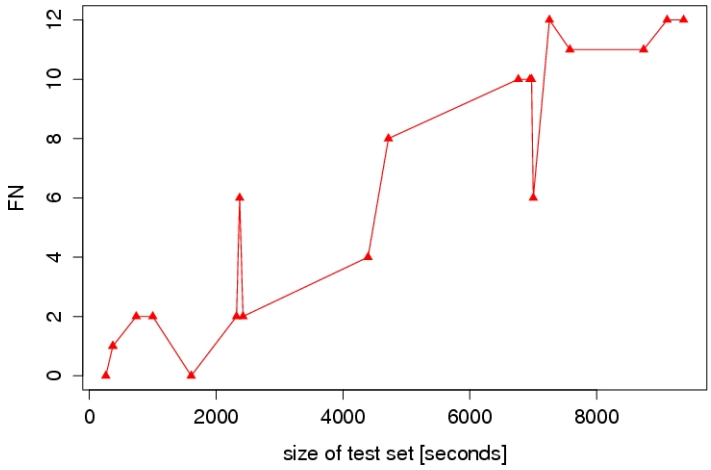


Figure 9.6: False negatives for fixed size of training set (7056 seconds) and varied size of test set with normal instances. FN roughly follows a linear trend.

While this first experiment was conducted with a fixed test set size, as a next step, the number of false negatives for larger sizes of the test set is investigated. Obviously the more data is tested, the more false negatives are likely to be reported. With the maximal available training set (7056 seconds), the size of the test set is varied. As indicated by Figure 9.6 the number of false negatives roughly follows a linear trend.

Since the absolute number FN depends on the size of the test set, the measure FN/h is introduced which gives the number of false negatives per hour. A crucial observation

for the applicability of the approach can be made from Figure 9.7. As the test set grows, FN/h becomes approximately constant.

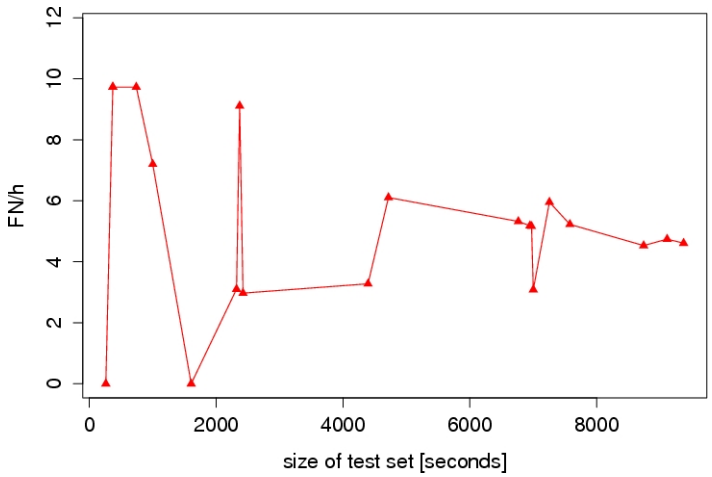


Figure 9.7: Ratio between false negatives and size of test set for a fixed size of the training set (7056 seconds) and a varied size of test set with normal instances.

9.2.1.1 Discussion

A number of important conclusions are drawn from the above experiments. The number of false negatives per hour becomes approximately constant for a big enough test set. This means that, for a given training set, an estimate of the expected number of false negatives can be deduced.

If during operation of the anomaly detection system, FN/h on unseen data significantly deviates from the figure determined during training, the training set has become non-representative. In that case the training set should be enhanced and the training period re-run.

9.2.2 Experiments with recordings from idle mode containing errors

After the initial experiments, the maximal available training set of 7056 seconds from Figure 9.5 was selected for further experiments on recordings with faults.

Faults were injected into the vehicle and several recordings in idle mode were used as a test set. 33 faults were injected in total: different spark plug leads were temporarily disconnected 23 times (fault #4 from Section 9.1), 5 positive and 5 negative offsets were injected into the engine coolant temperature sensor (fault #2). The test set contains 6 recordings summing up to 4342 seconds.

As in Section 9.2.1, the classification accuracy w.r.t. the size of the test set is measured. As an example, one of the recordings with faults injected by temporarily disconnecting the spark plug lead for 10 times is shown in Figure 9.8. The resulting errors are obvious in the "STFT" signal. All of the 10 errors were detected, two false negatives were reported.

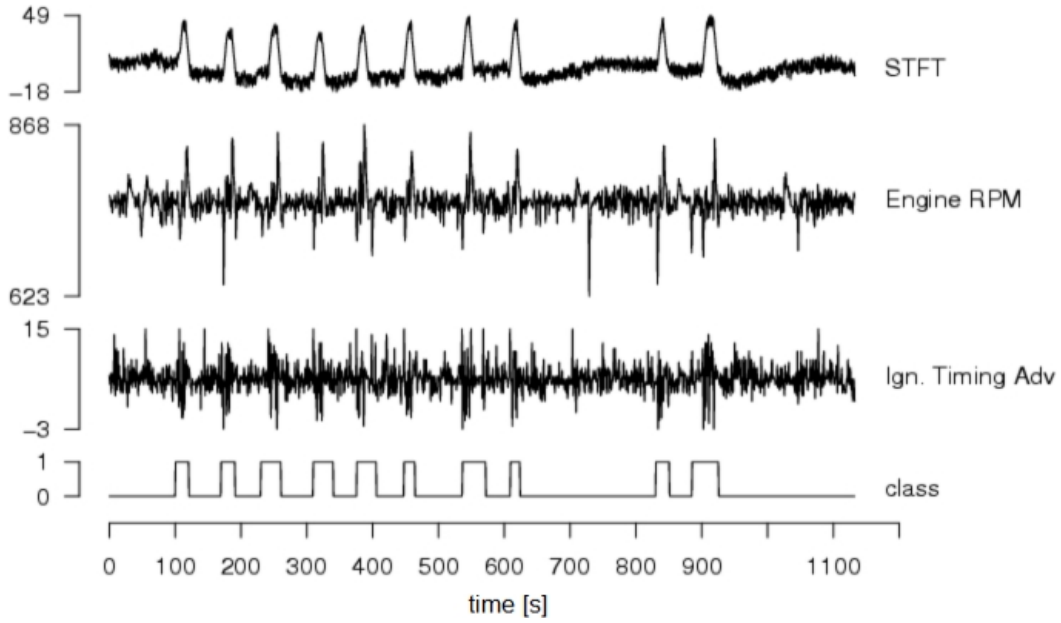


Figure 9.8: Recordings of vehicle in idle mode, with 10 injected faults. The faults are indicated by values of 1 for the label "class".

data set	$\|\mathcal{F}\|$	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	SVs	FN	FN/h	TN	TNR	precision
idle mode	3	1714s	3958s	54	7	6.3	31	93.9%	81.6%
idle mode	3	7056s	3958s	54	7	6.3	31	93.9%	81.6%

Table 9.3: Results with two training sets of different lengths on recordings in idle mode with 33 injected faults, where FN/h is the number of false negatives per hour.

The number of true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN) w.r.t. the size of the test set are shown in Figure 9.9 and the percentages TPR, FPR, FNR, and TNR are depicted in Figure 9.10.

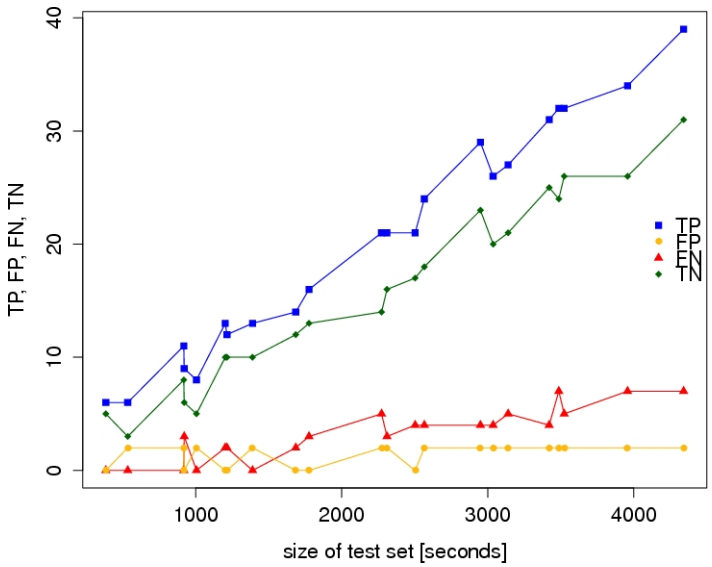


Figure 9.9: Classification results on a test set with 33 injected faults. TP, FP, FN, and TN for a training set size of 7056 seconds and a varied size of the test set with normal and abnormal data.

Table 9.3 shows the results for the size of the training set that was identified in the previous experiment to be the minimal training set and the full available training set. As can be seen, both training sets yield the same results, 93.9% of the faults were detected. The results are very good, which was expected, as the faults could easily be visually identified in the plot as shown in Figure 9.8.

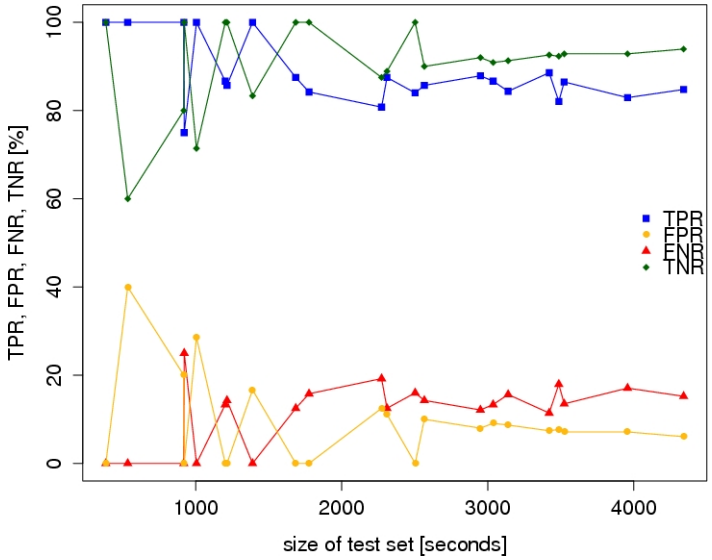


Figure 9.10: Classification results on a test set with 33 injected faults. TPR, FPR, FNR, TNR w.r.t. the size of the test set.

9.3 Experiments with error-free recordings from test drives

Further experiments were conducted with recordings from real test drives. Based on these recordings, the effect of different driving conditions, different drivers and different vehicles are investigated in this section.

During test drives, the 8 signals shown in Table 9.4 were recorded. The signal "load" is the engine's load, "coolant temp" holds the temperature of the engine coolant, and "STFT" (short term fuel trim) holds the injection pulse width, which keeps the air-fuel ratio optimal, i.e. the lambda value close to 1. The signal "MAP" is the manifold absolute pressure which is used to calculate the air mass flow rate, which in turn determines the fuel to be injected for optimal combustion. Furthermore, "rpm" is the engine revolution, "speed" is the vehicle's speed and "ign timing adv" (ignition timing advance) measures the angle of the piston position where the ignition takes place. Finally the value of "throttle" is the position of the throttle valve, which is directly proportional to the accelerator pedal position.

Signal short name	Signal full OBD name	unit	OBD PID
load	Calculated Load Value	%	04 hex
coolant temp	Engine Coolant Temperature	°C	05 hex
STFT	Short Term Fuel Trim (Bank 1)	%	06 hex
MAP	Intake Manifold Absolute Pressure	kPA	0B hex
rpm	Engine RPM	rpm	0C hex
speed	Vehicle Speed	km/h	0D hex
ign timing adv	Ignition Timing Advance (Cylinder 1)	°	0E hex
throttle	Absolute Throttle Position	%	11 hex

Table 9.4: Signals used for the experiments on recordings from test drives.

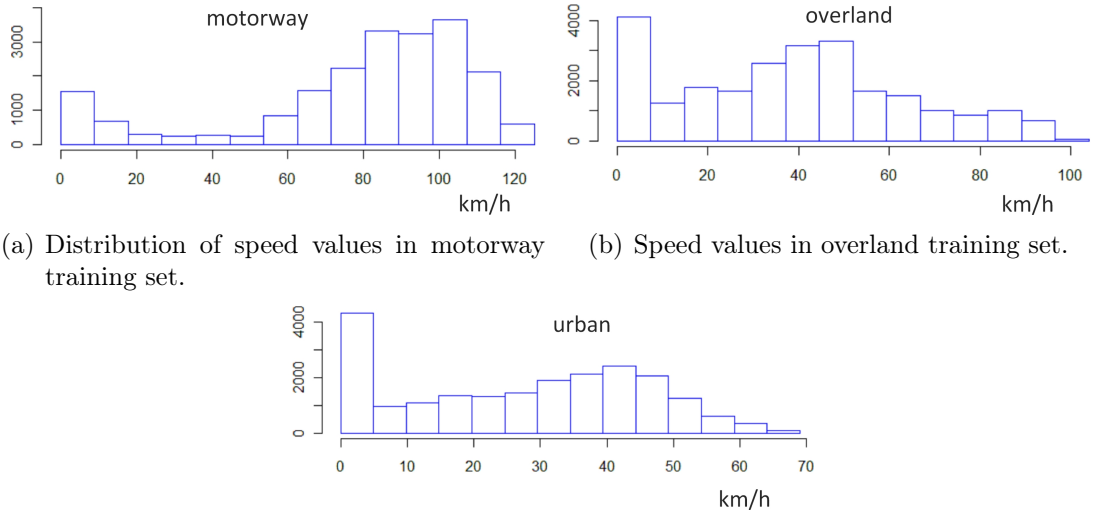
9.3.1 The effect of different driving conditions

The variability of test drive data is enormous due to, for instance, different road conditions, traffic conditions, or drivers. As a first step, the test drives are categorised into three groups based on the driving conditions. The main characteristics of the different driving conditions are shown in Figure 9.11.

1. *motorway*: These data sets contain test drives recorded on motorways. Their main characteristic is that the speed is predominantly within the range of 80 and 110 km/h.

2. *overland*: The overland data sets consist of recordings from test drives on B-roads, containing urban traffic.

3. *urban traffic*: These data sets were recorded while driving in a town or city. Low speed values occur frequently from stop-and-go traffic. The higher speed values are almost uniformly distributed up to a speed of 50 km/h.



(a) Distribution of speed values in motorway training set.

(b) Speed values in overland training set.

(c) Speed values in urban training set.

Figure 9.11: Distribution of the speed values in the used training sets from different driving conditions.

9.3.1.1 Motorway

In order to show the variability of the signals, an excerpt of a recording of an error-free test drive on a motorway with the 8 observed signals is shown in Figure 9.12.

The same set of experiments that was conducted on error-free recordings from idle mode in Section 9.2.1 is conducted on recordings from test drives on motorways. As a first experiment, the size of the training set is grown for a fixed error-free test set of 5408 s. The false negative rate is measured to determine how representative the training set is. In the second experiment, the maximal available training set with 20843 s is selected and the size of the test set is grown.

The number of false negatives FN w.r.t. the ratio between the training set size and the the test set size $\frac{\|\mathcal{A}\|}{\|\mathcal{B}\|}$ is shown in Figure 9.13. As in Section 9.2.1, this plot can be used to find the minimal training set.

Growing the test set for the maximal available training set, the number of false negatives increases as shown in Figure 9.14. As can be seen, there is an approximately linear

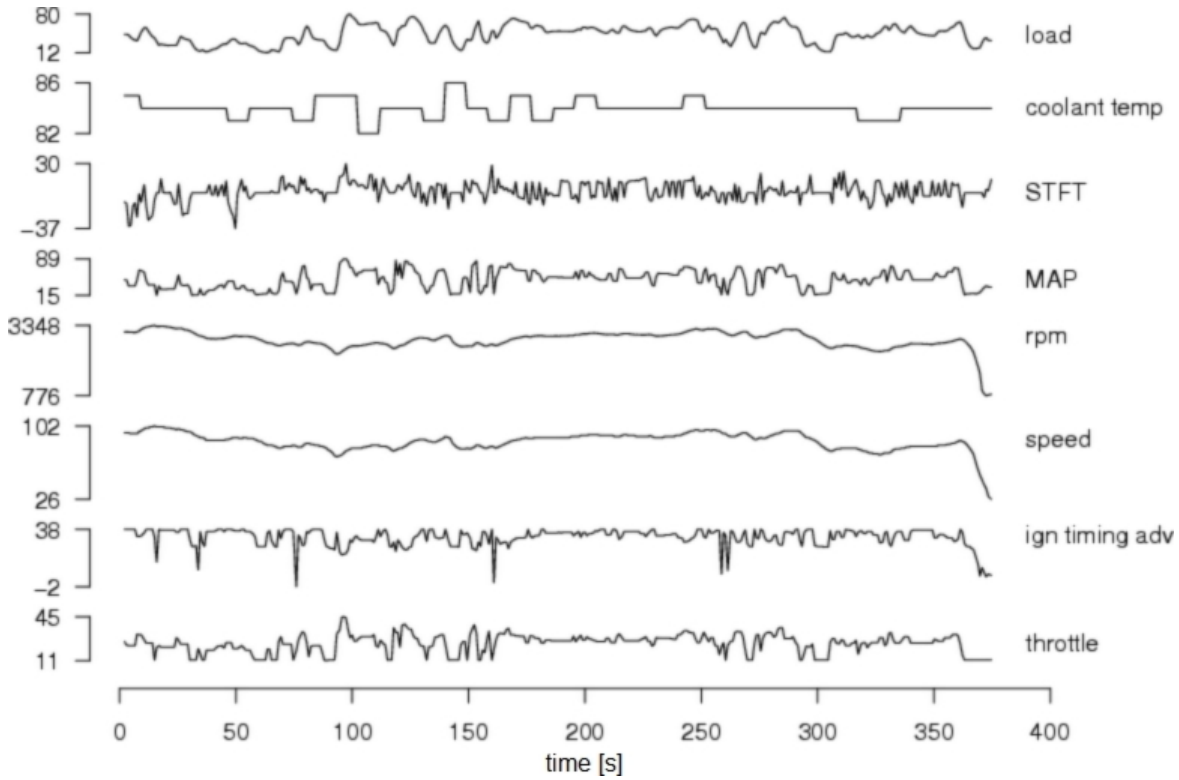


Figure 9.12: 6 minutes excerpt of a recording from a test drive on a motorway showing the 8 signals recorded during test drives.

trend. The number of false negatives per hour FN/h stays approximately constant for any size of the test set as shown in Figure 9.15.

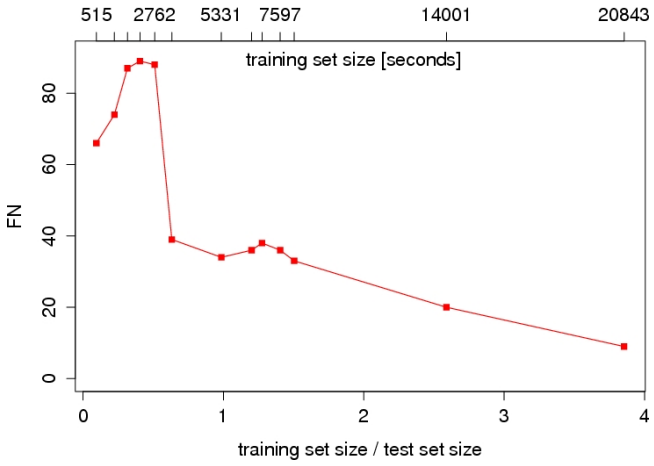


Figure 9.13: Recordings from motorway: The number of false negatives w.r.t. the ratio between the size of the training set and a fixed size of an error-free test set, where the size of the training set is shown by the second x-axis.

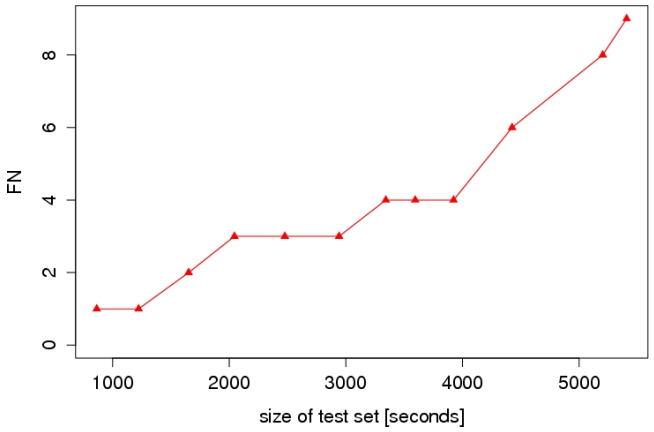


Figure 9.14: Recordings from motorway: False negatives for the selected optimal training set and a varied size of test set with normal instances.

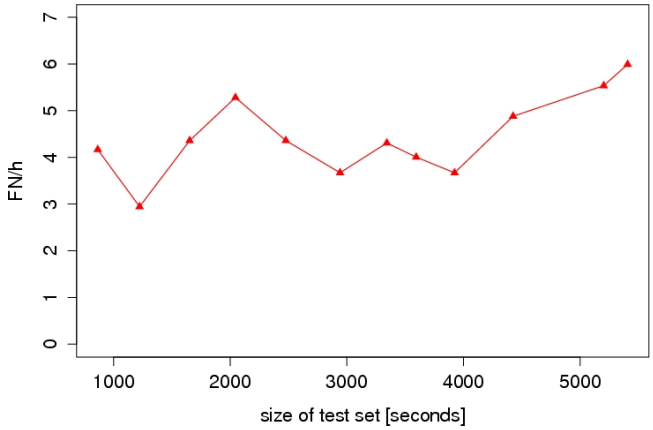


Figure 9.15: Recordings from motorway: The number of false negatives per hour FN/h is approximately constant for all sizes of the error-free test set.

9.3.1.2 Overland

The same experiments were conducted for recordings from overland drives. For a fixed test set size of 7010 seconds, the training set was varied and the FN plotted as shown in Figure 9.16. It can be seen that the number of false negatives stabilises at a low value for large training sets. For the maximal size of the training set, only one false negative is reported, which makes further plots, contemplating the number of FNs with a growing test set size, unnecessary. The value range would be between 0 and 1 false negatives.

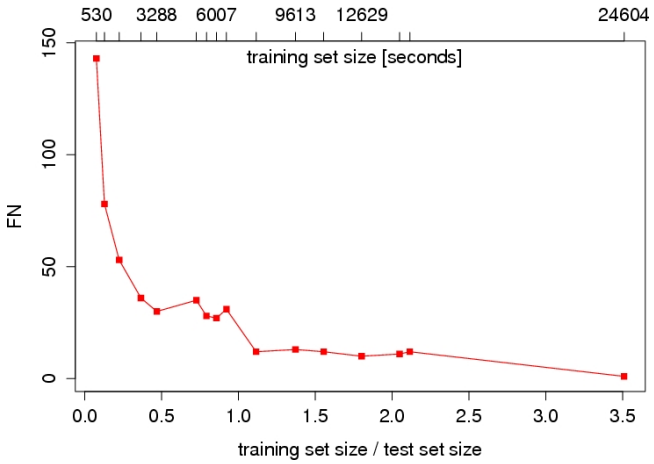


Figure 9.16: Recordings from overland drives: The number of false negatives w.r.t. the ratio between the size of the training set and a fixed size, error-free test set.

9.3.1.3 Urban traffic

For a fixed test set size and a growing training set, the number of false negatives w.r.t. $\frac{\|\mathcal{A}\|}{\|\mathcal{B}\|}$ for test drives recorded in urban traffic is shown in Figure 9.17. For large training sets, FN takes on low values for the 5703 seconds long test set.

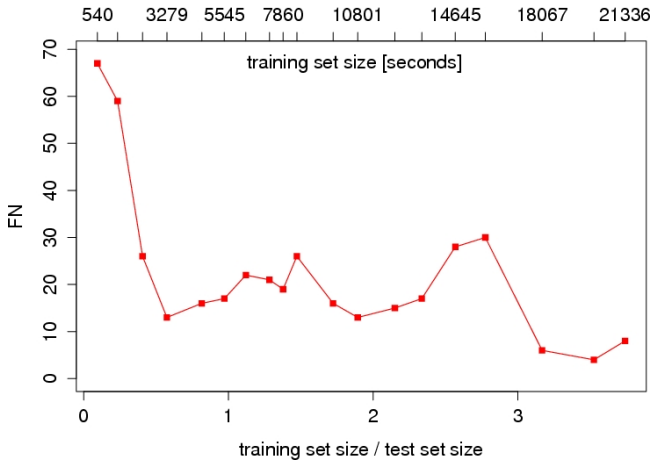


Figure 9.17: Urban traffic: False negatives for varied size of training set and fixed size of test set with normal data only.

9.3.1.4 Results on different driving conditions

The results on error-free test sets for the three driving conditions are shown in Table 9.5. The error rate, i.e. the number of false negatives, is very low for recordings from overland drives, which is an indication that the available training set is representative.

In real test drives, one recording may contain sections with all driving conditions. Segmenting and assigning the segments in the recordings to one of the driving conditions prior to anomaly detection would be a costly, error-prone process. Hence, a detection system should be able to classify arbitrary recordings.

training,test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	SVs	FN	FN/h	avg W	data points
motorway,motorway	20843s	5408s	177	9	6.0	8	1.4%
overland,overland	24604s	7010s	219	1	0.5	10	0.1%
urban,urban	21336s	5703s	189	3	1.9	26	3.6%

Table 9.5: Results for training and testing on the same driving condition with error-free test sets. The column "FN/h" holds the number of false negatives per hour.

Having looked at the driving conditions individually, the next step is to investigate what happens if a training set does not represent all driving conditions properly. The effect of training on one driving condition and testing on data from a different one is shown in Figure 9.18, where the percentages of falsely classified data points are given. For example the first column shows the result for training and testing on recordings from motorway traffic. It is expected that the best results occur for training and testing with the same driving conditions, i.e. the first column in the first group of columns, the second column in the second group and so forth.

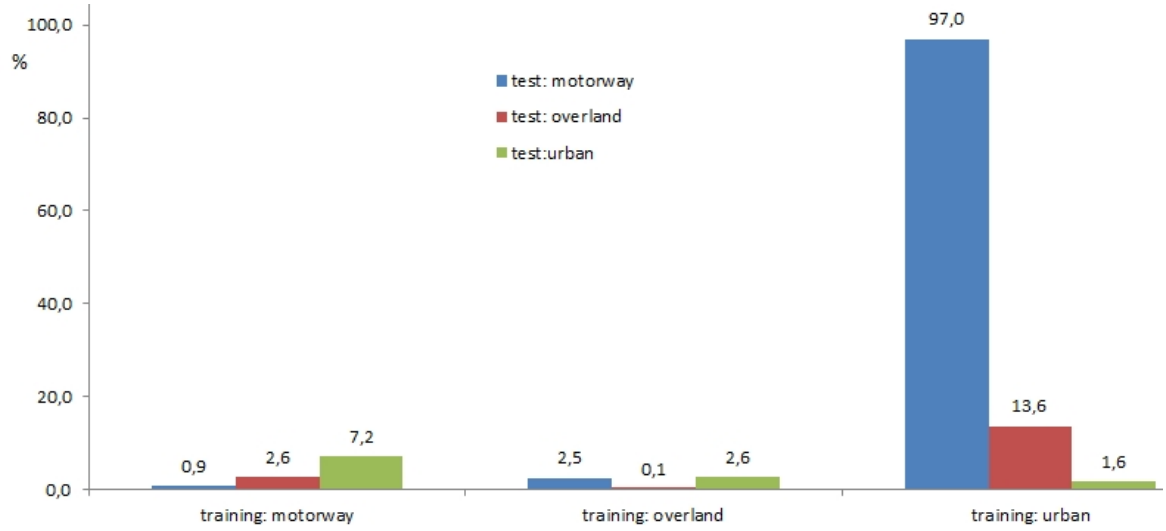


Figure 9.18: Percentage of data points falsely reported as abnormal for training and test set from the same and from different driving conditions.

Having seen that the results are sensitive to the driving conditions, the question arises, if the training set becomes more representative by combining training sets from different driving conditions. This is investigated in the next experiment.

As shown in Table 9.6, the results for the motorway and the urban test sets have notably improved compared to the results with individual training sets given in Table 9.5. Even though the result for the overland data set is slightly weaker, having increased from 1 to 2 false negatives, it is considered to be unchanged at such a low number of false negative. In conclusion to this, the observation is that the training set has indeed become more representative by combining the individual training sets.

training,test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	SVs	FN	FN/h	avg W	data points
all,motorway	63631s	5408s	302	0	0.0	0	0.0%
all,overland	63631s	7010s	302	2	1.0	10	0.3%
all,urban	63631s	5703s	302	3	1.9	11	0.6%

Table 9.6: Results with one combined training set on error-free test sets.

As a consequence, a new recording can be tested on this combined training set, without having to determine whether it is motorway, overland, or urban traffic.

9.3.2 The effect of different drivers

In the previous experiments the results on different driving conditions were compared. The recordings used as the training and test sets were obtained from test drives from the same driver. In practice, when testing a vehicle fleet the scenario is not one driver testing one vehicle, but rather multiple drivers testing multiple vehicles.

So it is necessary to study the effect of different drivers. In order to negate the effects caused by different driving conditions, the experiments are conducted on recordings from urban traffic with test vehicle "tw1". The drivers are given in Table 9.7, where all previous experiments were conducted on data from driver "dr1".

The results for different drivers are shown in Table 9.8. Training and testing with the same driver, as given in the first row, yields by far the best results with 4.4 FN/h. Training on recordings from one driver and testing on recordings from a different driver yields weaker results.

id	age	gender
dr1	38	male
dr2	30	male
dr3	25	male

Table 9.7: Drivers that conducted the test drives used in the experiments.

training,test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	SVs	FN	FN/h	avg W	data points
dr1,dr1	21336s	5703s	189	3	1.9	26	3.6%
dr1,dr2	21336s	6486s	189	14	7.8	7	1.6%
dr1,dr3	21336s	5701s	189	45	28.4	10	8.2%
dr2,dr1	11662s	5703s	128	18	11.4	28	8.9%
dr2,dr3	11662s	5701s	128	46	29.0	7	6.4%

Table 9.8: Results on error-free test sets with training and testing on recordings from different drivers.

As shown by the results, the effect of the driver can be significant, which is clearly recognisable from the results when testing with "dr3". The results are significantly weaker compared to testing with "dr1" and "dr2". The reason is, that the driving behaviour of "dr3" differs from "dr1" and "dr2". As a consequence the training set does not represent "dr3" properly.

To investigate the reasons, the visual data mining techniques from Chapter 4 can be used. From the 8 recorded signals, the engine rpm is shown w.r.t. the vehicle speed in the scatter plot in Figure 9.19. The five lines with different gradients represent the vehicle's five gears. It can be seen that some of the false negatives were caused by "dr3" driving the gears up to a higher engine rpm compared to the drivers included in the training set (marked by the three black frames).

There are two ways to cope with the effect of different drivers. One way is to train and test with recordings from the same driver. This is feasible in the research and development stage of a vehicle, when engineers drive the vehicles themselves. When testing a vehicle fleet, the test drives are conducted by multiple drivers. In that case the training set should contain test drives from various drivers with different driving behaviour to be representative.

The question is, whether the system can be used on recordings from a driver not included in the training set. The expectation is that by integrating multiple drivers with different driving behaviours in the training set, a training set can be created that is representative for most drivers. First, recordings from drivers included in the training set are tested, in further experiments the driver is not included in the training set. The results are shown in Table 9.9.

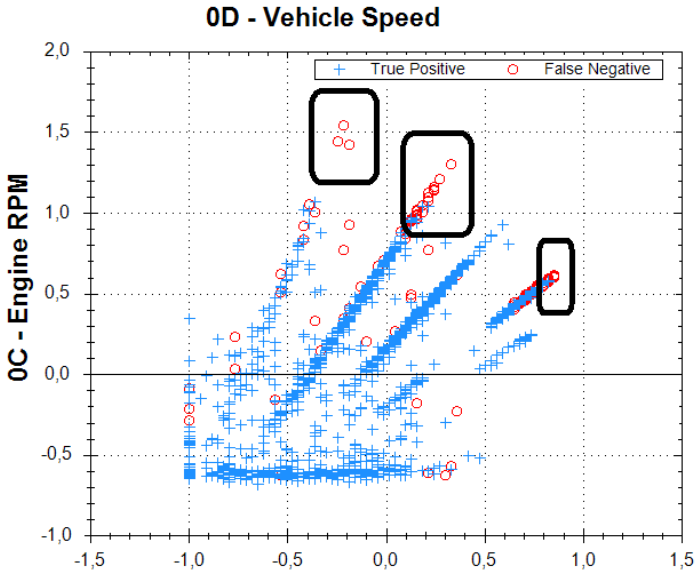


Figure 9.19: The engine rpm w.r.t. the vehicle speed with a test set from "dr3" and a training set from "dr1" in normalised feature space (false negatives: red circles).

training,test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	SVs	FN	FN/h	avg W	data points
dr1+dr2,dr1	34820s	5703s	254	4	2.5	21	1.5%
dr1+dr3,dr1	28859s	5703s	210	1	0.6	5	0.1%
dr1+dr2,dr3	34820s	5701s	254	6	3.8	8	0.87%
dr1+dr3,dr2	28859s	6486s	210	2	1.1	5	0.2%

Table 9.9: Results on error-free test sets with training sets from different drivers.

Training on recordings from "dr1" and "dr2" and testing on data from "dr3" is shown in the third row of Table 9.9. It can be seen that the results have significantly improved from 28.4 FN/h given in the third row in Table 9.8 to 3.8 FN/h given in the third row in Table 9.9. In conclusion to this, it is possible to use the detection system with test data from a driver not previously included in the training set.

The drivers involved in these experiments were non-professional drivers that were told to drive in their normal way. Professional test drivers can be told to mimic different driving behaviour, so a representative training set can be obtained from a limited number of test drivers.

Id	Brand	Date of manufacture	Engine displacement	Engine power	Weight	OBD physical layer
tw1	Renault Twingo	2002	1149 ccm	43 kW	895 kg	K-Line
tw2	Renault Twingo	2002	1149 ccm	43 kW	895 kg	K-Line
tw3	Renault Twingo	2011	1149 ccm	55 kW	1019 kg	CAN
tw4	Renault Twingo	2012	1149 ccm	55 kW	994 kg	CAN

Table 9.10: Vehicles used for the experiments.

9.3.3 The effect of different vehicles

At this point different driving conditions and different drivers were considered for recordings from test vehicle “tw1”. In practice, various vehicles are driven for testing purposes. So, analogous to studying the impact of the driver, an experiment investigating the influence of the vehicle is indispensable. The experiment is conducted on recordings from urban traffic with driver “dr1” and various vehicles.

The used test vehicles are shown in Table 9.10. The vehicles “tw1” and “tw2” have identical engines and are of the same model series, manufactured in the same year. There are slight differences, e.g. “tw1” has power assisted steering while “tw2” has not. The two vehicles “tw3” and “tw4” are vehicles from the successor model series with a different engine and chassis.

The results are shown in Table 9.11. Regardless, if training and testing is done on the same vehicle, as shown in the first and fourth row, or on different vehicles of the same model series, as in the second and third row, the error rates of the results should show no significant difference. The results shown in the first four rows of Table 9.11 indicate that the effect of testing with different vehicles from the same model series is not significant. Training with “tw2” and testing with “tw1” yields weaker results compared to the other three rows. The reason is, that the training set of “tw2” is not large enough to be representative.

For training with vehicles from one model series and testing with vehicles from the successor model series the results are significantly different as indicated by the last four rows in Table 9.11.

training,test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	SVs	FN	FN/h	avg W	data points
tw1,tw1	21336s	5703s	193	7	4.4	12	1.49%
tw1,tw2	21336s	6020s	193	5	3.0	6	0.50%
tw2,tw1	12496s	5703s	169	34	21.5	26	15.5%
tw2,tw2	12496s	6020s	169	5	3.0	5	0.41%
tw1,tw3	21336s	1738s	193	15	31.1	7	6.6%
tw1,tw4	21336s	2525s	193	56	79.8	10	23.9%
tw2,tw3	12496s	1738s	169	44	91.1	9	23.2%
tw2,tw4	12496s	2525s	169	61	87.0	19	47.5%

Table 9.11: Results on error-free test sets with different vehicles.

9.3.4 Discussion

Recordings from test drives are much more variable than recordings from the vehicle in idle mode. After having individually investigated the effect of different driving conditions, different drivers, and different vehicles, the results are summarised as follows:

- the driving conditions have a major impact on the number of false negatives as shown in Table 9.5
- the driver has a significant effect as shown in Table 9.8
- if using several vehicles from the same models series, no significant difference is expected as can be seen in Table 9.11

The following conclusions are drawn from the experiments: An ideal training set

1. has to be large enough to contain common driving conditions, and different traffic situations
2. should contain recordings from different test drivers
3. may contain recordings from various vehicles of the same model series, if available

9.4 Experiments with test drives containing errors

The previous experiments with recordings from error-free test drives studied how to determine if a training set is representative in the absence of abnormal data and how to estimate the expected false negative rate. Furthermore the effects of the driving condition, the driver, and the vehicle on the representativeness of the training set were investigated.

This section tests the anomaly detection system with the scenario encountered in practice: training on error-free data and testing on unknown data that potentially contains anomalies. The detection system's accuracy on test drives with injected faults is measured.

9.4.1 Results on recordings from different driving conditions

This section investigates the results for test drives from different driving conditions. Training and test sets were recorded with vehicle "tw1" and driver "dr1".

Five test drives recorded on a motorway with a total of 21 injected faults were available. During the first of the drives, the spark plug lead was temporarily removed 4 times (fault #4 from Section 9.1). In the second, third and fourth recording, a cylinder's injector nozzle was temporarily switched off several times (fault #3). In the fifth test drive, the sensor measuring the engine coolant temperature was manipulated 6 times (fault #1 and #2).

The test set with overland drives contains five test drives with 42 injected faults. In the first two test drives, the spark plug lead was temporarily removed (fault #4) 10 times in each case. In the third and fourth test drive, one cylinder's injector nozzle was temporarily switched off to cause misfiring, 9 and 7 times respectively (fault #3). In the fifth recording, the temperature sensor was manipulated 6 times (fault #1 and #2).

From urban traffic eight test drives were used, containing 13 faults in total. The injector nozzle was temporarily switched off 10 times (fault #3) and the engine coolant temperature sensor was temporarily manipulated 3 times (fault #1 and #2).

As a first step, the system was trained and tested with recordings from one driving condition, i.e. motorway, overland, or urban traffic. The results are given in the first three rows in Table 9.12.

Roughly between half and three quarters of the faults were detected. The highest detection rates are achieved on data from urban traffic. This is due to the fact that faults #3 and #4 become more obvious when varying the engine's load, which is more likely to occur during a drive in urban traffic. This explains the lower detection rate for motorway traffic, due to mostly steady traffic on motorways. It is noticeable that FN/h is higher than in the experiments on error-free data in Table 9.5. This is an indication, that the training set for overland and urban data is not fully representative.

For the subsequent experiments the system was trained on the recordings from all driving conditions, the same training set used in Table 9.6 in Section 9.3.1. The last four rows in Table 9.12 show the results for these experiments. With the combined training set, FN/h has significantly decreased compared to the experiments with individual training sets shown in the first three rows.

training,test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	FN	FN/h	TN	TNR	precision
motorway,motorway	20843s	4845s	9	6.7	9	42.9%	50.0%
overland,overland	24604s	12076s	16	4.8	31	73.8%	66.0%
urban,urban	21336s	7224s	21	10.5	10	76.9%	32.3%
all,motorway	63631s	4845s	0	0.0	10	47.6%	100%
all,overland	63631s	12076s	10	3.0	27	64.3%	73.0%
all,urban	63631s	7224s	4	2.0	9	69.2%	69.2%
all,all	63631s	24145s	14	2.1	45	59.2%	76.3%

Table 9.12: Results for motorway, overland, and urban test drives.

The results for the three recordings given as examples for the four types of faults in Section 9.1 are shown in more detail. These three recordings are contained in the test set of the experiment given in the fifth row in Table 9.12, with a training set consisting of recordings from all driving conditions.

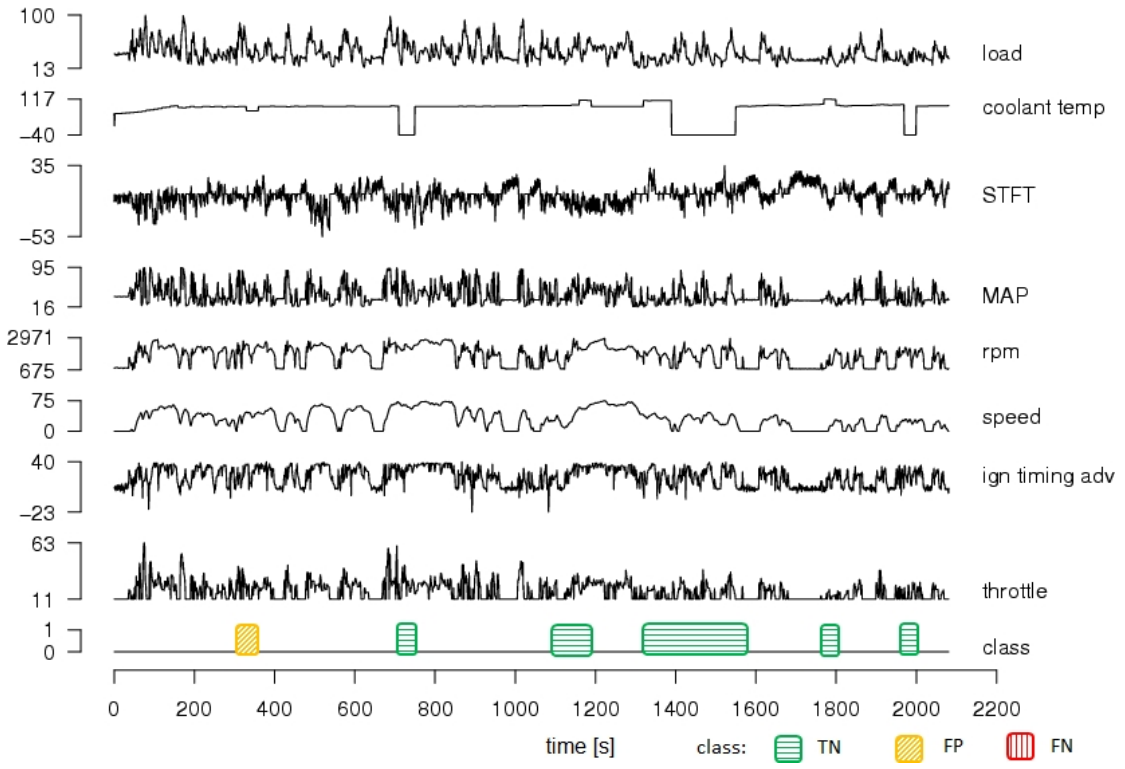


Figure 9.20: Example of classification results for faults injected during an overland drive of 35 minutes by manipulating the temperature sensor. The results are marked with frames (TN: green, FP: yellow).

Figure 9.20 shows the classification results for injection of fault #1 and #2, where the temperature sensor was manipulated. The first injected fault was not detected since the negative temperature offset leads to values within the valid range. Since the relationship to further signal is not violated, this fault is not detectable. All further faults lead to temperature values out of the range present in the training set and were reliably detected.

Furthermore, the test drive shown in Section 9.1 containing fault #3 was tested. The injector nozzles of the first or second cylinder were temporarily switched off 9 times (fault #3), causing misfiring. Seven of the nine faults were correctly detected. The results are shown in Figure 9.21, where the detected anomalies (true negatives) are highlighted with green frames. Six subsequences were falsely reported as anomalies

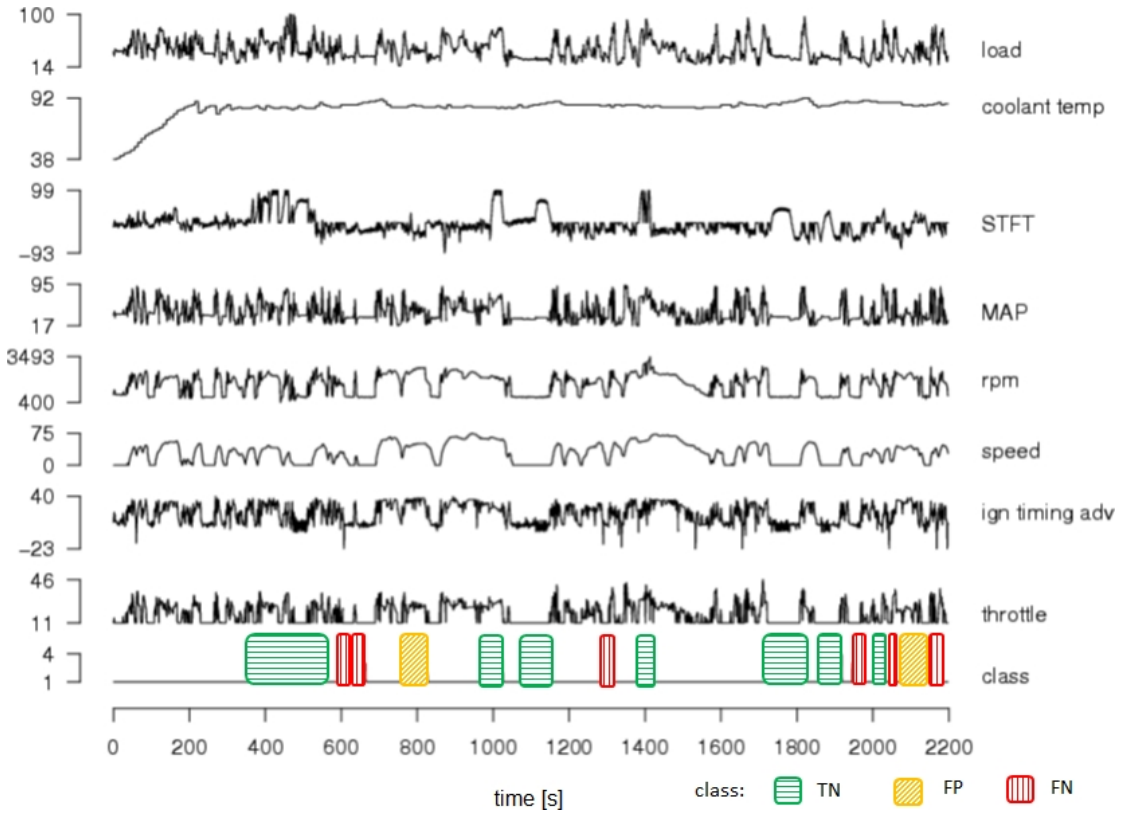


Figure 9.21: Classification results for a 35 minutes long recording of an overland drive where a the injector nozzle was manipulated 9 times. The results are marked with frames (TN: green, FP: yellow, FN: red).

(false negatives), indicated by red frames and two faults were not detected (false positives), shown by yellow frames.

Using the anomaly detection system in practice, it works on unknown, unlabelled data sets. Consequently, the system does not report true and false negatives. What the expert rather sees for this recording, are 13 subsequences reported as anomalies. The expert will investigate the reported subsequences and 7 of the 13 will point the expert to a fault in the vehicle.

In Figure 9.22 the results on a one hour test drive are shown, where the spark plug lead was temporarily disconnected while the vehicle was standing still (fault #4). From the 10 injected faults, 8 were detected. None of the signals is out of the valid value

range, the faults were detected solely due to violations of learnt relationships between signals. No anomalies were falsely reported.

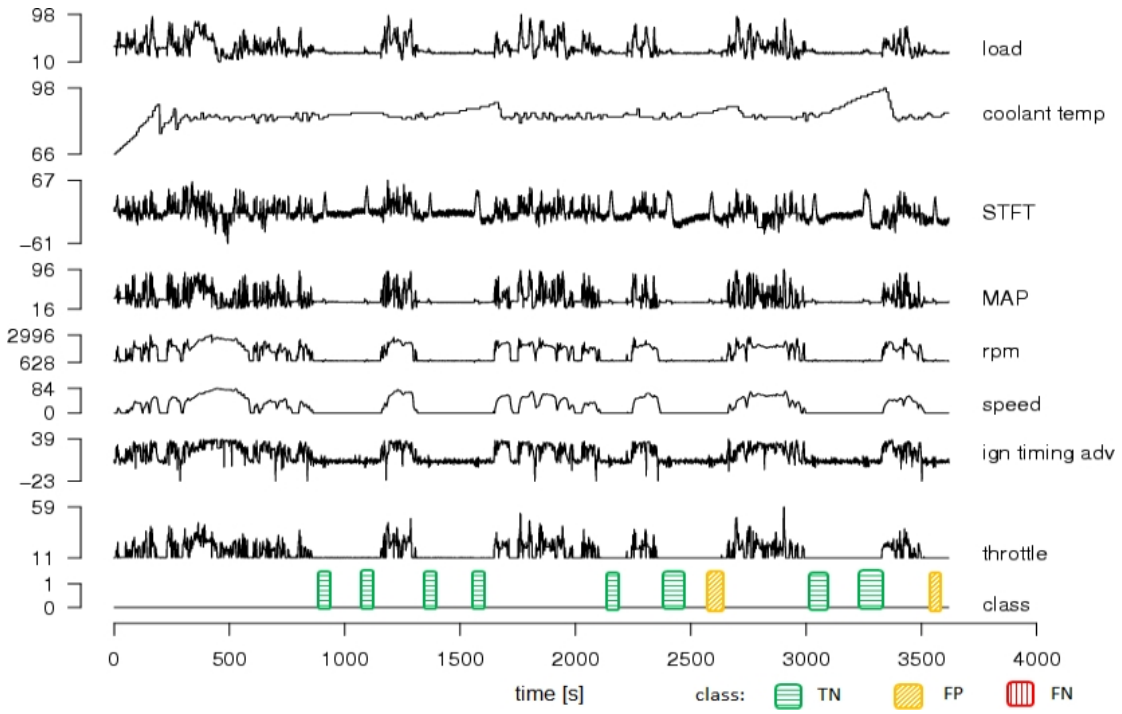


Figure 9.22: Classification results for an overland drive of one hour where 10 faults were injected by temporarily interrupting the spark plug lead. The results are marked with frames (TN: green, FP: yellow).

Summarising, from the anomalies introduced in Section 3.3, it was shown that type 1 and type 3 can be detected. With the training set containing recordings from motorway, overland, and urban drives, for the accumulated test set (last row in Table 9.12) approximately 60% of the injected faults were detected, while the percentage of faults in the result set of reported anomalies was 76%. This shows the effectiveness of the approach, since an expert will be able to detect the faults in the result set without having to investigate many falsely reported anomalies.

9.4.2 Results on recordings from different drivers and vehicles

While in the previous section, training and test set were taken from the same vehicle and the same driver, the experiments in this section cover the scenario that is encountered when testing vehicle fleets. Recordings from different vehicles and different drivers in urban traffic are used as the training and test sets.

Three different test sets are used. The first test set ("tw1, dr1") contains 8 test drives from vehicle "tw1" and driver "dr1", where the injector nozzle was temporarily switched off 10 times (fault #3) and the engine coolant temperature sensor was manipulated 3 times (fault #1 and #2). The second test set ("tw2, dr1") contains 8 recordings from driver "dr1" and vehicle "tw2" with 6 fault injections into the engine coolant temperature (fault #1 and #2) and 4 into an injector nozzle (fault #3). The third one ("tw2, dr2") consists of 7 test drives from driver "dr2", with 3 manipulations of the engine coolant temperature sensor (fault #1 and #2) and 4 manipulations of an injector nozzle (fault #3).

The first row in Table 9.13 shows the result for training and testing with the same driver, the second row for testing with data from a different driver. While the detection rate is very good with 76.9% of the faults being detected, the false negative rate is high when training with driver "dr1" and testing with "dr2". This issue has been discussed in Section 9.3.2, the results can be improved by training on recordings from various drivers.

The result for testing with recordings from a different vehicle is given in the third row. The achieved detection rate of 50% is considered as good and the low false negative rate of 2.4 FN/h keeps the analysis effort low.

The fourth row in Table 9.13 shows the result when testing with a driver and vehicle, that are both not included in the training set. By using a training set from two different drivers ("tw1, dr1+dr3"), the number of false negatives can be reduced from 13 to 3, while the true negative rate remains unchanged. This result is given in the last row showing a percentage of detected anomalies of 57.1%. In combination with the low false negative rate of 1.5 FN/h this shows that the proposed approach is applicable to

the testing of vehicle fleets, where recordings from a specific vehicle or driver are not necessarily included in the training set.

training	test	$\|\mathcal{A}\|$	$\|\mathcal{B}\|$	FN	FN/h	TN	TNR	precision
tw1, dr1	tw1, dr1	21336s	7224s	21	10.5	10	76.9%	32.3%
tw1, dr2	tw1, dr1	11662s	7224s	38	18.9	10	76.9%	20.8%
tw1, dr1	tw2, dr1	21336s	7616s	5	2.4	5	50.0%	50.0%
tw1, dr1	tw2, dr2	21336s	7109s	13	6.6	4	57.1%	23.5%
tw1, dr1+dr3	tw2, dr2	34820s	7109s	3	1.5	4	57.1%	57.1%

Table 9.13: Results with training and test sets from different vehicles and drivers.

In Figure 9.23, one test drive with "tw2" in urban traffic from the experiment in the last row of Table 9.13 is presented in more detail. Fault #3 was injected four times. Three out of four anomalies were detected, marked in Figure 9.23 with green frames, the undetected anomaly is marked with a yellow frame. No false negatives were reported.

The system reports 3 abnormal subsequences. After analysis of the reported anomalies by the expert, all 3 anomalies turn out to be faults in the vehicle, one fault remains undetected.

Summarising, the results show the effectiveness of the approach. Roughly between 50% and 75% of the injected faults were detected, while the time for the analysis is kept reasonably low due to a moderate number of false negatives.

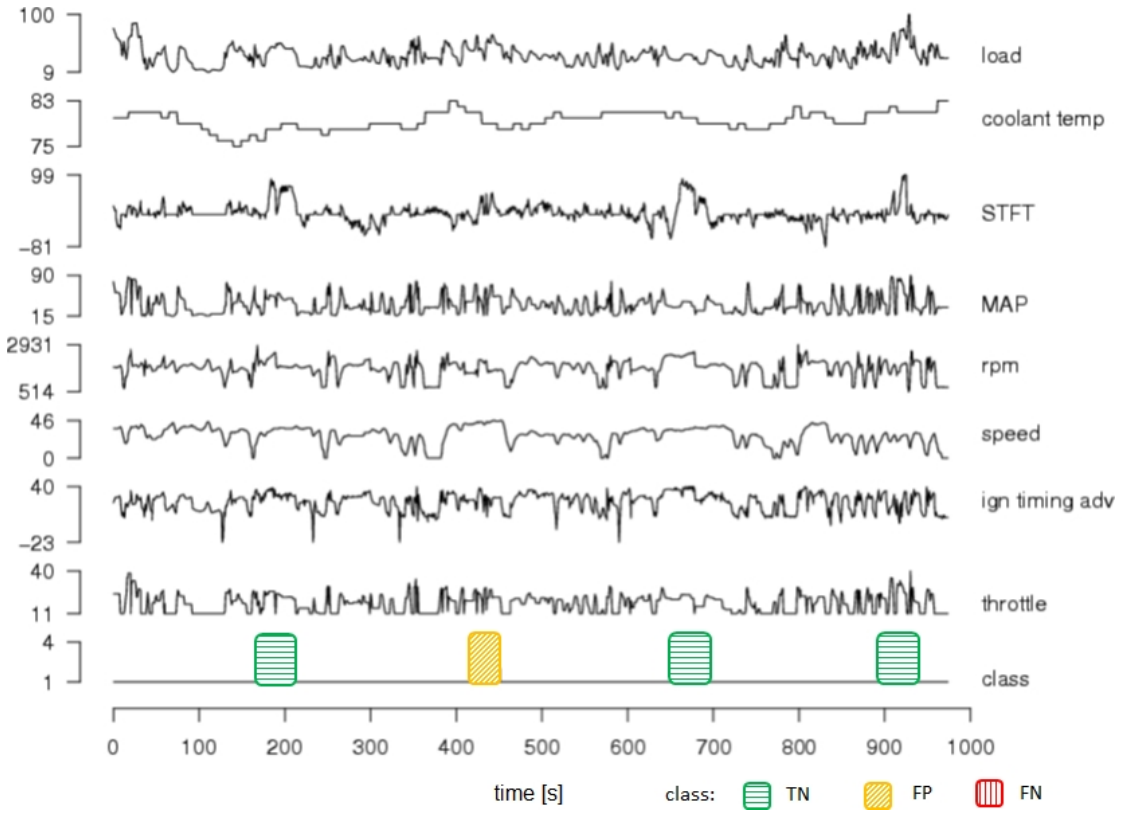


Figure 9.23: Classification results for a recording of 15 minutes from a driver and vehicle that are both not contained in the training set. The results are marked with frames (TN: green, FP: yellow).

9.5 Evaluation

In this section, the approach is further evaluated based on the experimental results. The scalability in terms of the size of the training set and the effect of the inherent parameter W, determining the lengths of the subsequences, are discussed. Following that, the effectiveness of the approach is quantified with respect to the aims given in Section 1.4.

9.5.1 Scalability of the approach

The classification of recordings is fast and is linear in the size of the test set as testing is done sequentially. The wall clock time required for training including parameter optimisation for the training set of Section 9.2 is shown in Figure 9.24. In general, the complexity for training of support vector machines is in the range of $O(n^2)$ and $O(n^3)$ as stated in (Bordes et al., 2005). The observed complexity is approximately quadratic in the number of training instances.

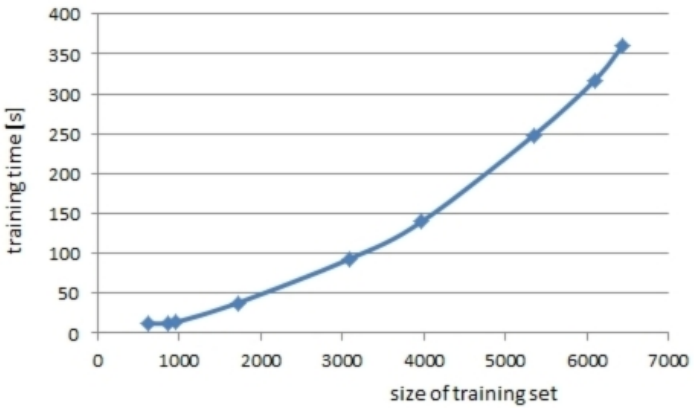


Figure 9.24: Required time for training w.r.t. the size of the training set.

As an example the time for training and test were measured on a current standard PC for the data set from Table 9.6 in Section 9.3.1.4, which contains 8 signals and 63631 seconds (17.5 hours) of recordings. The training period takes approximately 20 hours. Testing recordings of 20000 seconds (5.5 hours) takes only 2 seconds.

9.5.2 Effect of the length of the subsequences

The parameter W determines the length of the subsequences formed by SVDDSUBSEQ. It specifies to which extent the local neighbourhood of data points is taken into account. By its implicit filtering functionality it also prevents individual data points from being falsely reported as abnormal.

In the absence of abnormal test data, it is recommended to set the window length to a value that yields satisfactory false negative rates. If knowledge about the expected minimal length of a fault exists, the chance of detecting faults can be increased by setting the window length to values not greater than half the length of the fault. This way at least one subsequence holds data points that are all abnormal.

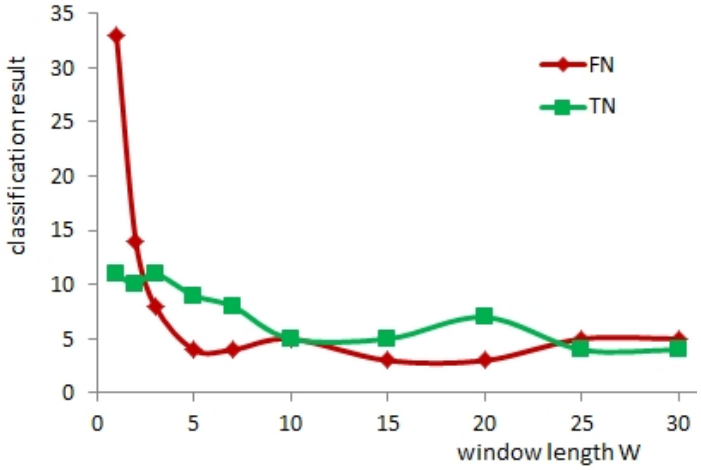


Figure 9.25: Classification results for different lengths of the subsequences.

However, as shown in Figure 9.25 the parameter W has no dramatic effect on the classification accuracy. For very small window sizes of 1 to 3 data points the number of false negatives is high, since the filtering effect of the window is not present. For larger window sizes FN and TN decrease gradually, showing that the range of a good window size is wide.

9.5.3 Quantification of the effectiveness of the approach

In order to quantify the effectiveness of the approach, eq. (1.1) from Section 1.4 is applied on the results with training and test set containing recordings from all driving conditions given in the last row in Table 9.12 in Section 9.4.1.

The number of detected faults on a set of recordings D_{all} was given in Section 1.4 as

$$N_{detected\ faults} = p(fault|d) * \frac{T_{budget}}{T_{diagnosis}} \tag{9.1}$$

where $p(fault|d)$ is the probability of a fault in one data set d, T_{budget} is the available time budget, and $T_{diagnosis}$ is the time required for the diagnosis of one data set d.

Two aims were formulated in Section 1.4:

1. decrease the time $T_{diagnosis}$ required for the analysis of one data set d

2. increase the probability $p(fault|d)$ of a fault in one data set d

The degree of improvement of $T_{diagnosis}$ by the introduction of the visual data mining techniques in Chapter 4 is hard to measure since it depends on the expert's knowledge and the quality of the currently used tools. In this evaluation it is assumed that the diagnosis is speeded up by a factor of 2, i.e. $\frac{T_{diagnosis_{current}}}{T_{diagnosis_{new}}} = 2$, where "current" refers to the currently used techniques and "new" refers to the approach proposed in this Thesis.

The increase of the probability $p(fault|d)$ is taken from the results in Table 9.12. The test set D_{all} is 24145 seconds long with 76 faults present. With a length of 30 seconds for each data set d, the expert has to investigate 805 portions of data in D_{all}. The result set $D_{potential\ errors}$ given in the experiment in Section 9.4.1 consists of 59 data portions d and 45 faults are present.

The question is posed, how much time would the expert have to invest with current techniques to detect the same number of faults detected in the experiments with the new approach, or in other words which speedup factor is achieved. The speedup factor is calculated by setting $N_{detected\ faults_{current}} \stackrel{!}{=} N_{detected\ faults_{new}}$:

$$N_{detected\ faults_{current}} \stackrel{!}{=} N_{detected\ faults_{new}} \tag{9.2}$$
$$p(fault|d_{all}) * \frac{T_{budget_{current}}}{T_{diagnosis_{current}}} = p(fault|d_{potential\ errors}) * \frac{T_{budget_{new}}}{T_{diagnosis_{new}}}$$

which is reformulated as

$$\frac{T_{budget_{current}}}{T_{budget_{new}}} = \frac{T_{diagnosis_{current}}}{T_{diagnosis_{new}}} * \frac{p(fault|d_{potential\ errors})}{p(fault|d_{all})} \tag{9.3}$$
$$speedup = \frac{1}{0.5} * \frac{\frac{45}{59}}{\frac{76}{805}} = 16$$

As can be seen from eq. (9.3), a significant improvement is achieved with the new approach. A speedup factor of 16 is achieved with the current approach on the given training and test set. This number is likely to be orders of magnitude higher in practice, since due to the injection of faults, the number of faults in the used test set is higher than expected in practice. Obviously the improvement in the number of detected faults by keeping the time budget fixed can be calculated likewise.

9.6 Conclusion

This chapter showed experimental results on recordings from vehicles. Starting with recordings from idle mode, the problem was gradually made more complex by adding different driving conditions, different drivers, and different vehicles.

A way was introduced to find a representative training set in the absence of abnormal test data by growing the size of the training set and monitoring the false negative rate.

Furthermore, the effect of different constitutions of the training set was investigated. The best results are achieved when training and test set are from the same driver and same vehicle.

The results are weaker when using training and test data from different drivers. It was shown, that the approach can be used to test recordings from drivers that are not included in the training set, by training on recordings from different drivers. Furthermore it was shown that using different vehicles of the same model series does not have a significant impact on the results. Conclusively, the results show that the approach can improve fault detection in recordings from test drives.

As all classification systems that solely learn from training sets, the proposed classifier SVDDSUBSEQ suffers from both types of classification errors: undetected anomalies (false positives) and falsely reported anomalies (false negatives). For that reason, the reported anomalies need to be investigated by an expert to cope with the false negatives. The visual data mining techniques introduced in Chapter 4 have proven very effective. The anomaly detection system proposed in this Thesis is therefore a semi-autonomous approach according to the categorisation given in Section 3.5.

It is in the nature of machine learning systems, that the quality of the results depends on the quality of the training set. If it is not possible to obtain a representative training set, the number of false negatives will be high. In that case the anomaly detection system is still useful in several ways:

1. Frequent or permanent faults can be detected by just monitoring the number of reported anomalies. If the number differs significantly from the expected FN rate, the test set is likely to contain anomalies.

2. A high number of false negatives is an indication that the training set is not representative. The false negatives can be used to identify why the training set is not representative and enhance the training set with missing data.

3. With the anomaly score specified in Section 7.4, the result set can be ranked and the subsequences with the highest anomaly scores can be investigated.

4. Even if the number of false negatives remains at a high level, the chance of detecting a fault in the result set is much higher than randomly selecting recordings to be investigated as described in Chapter 1.

While the experiments in this chapter were conducted on data related to the engine-management system, the methods discussed can be applied to data from different vehicle subsystems like braking systems, car infotainment, or driver assistant systems. In some cases, neither the driver nor the vehicle might have an impact on the classification accuracy, but rather different extra equipment of the vehicle, or different levels of equipment like base or premium versions of a component. Experiments, analogous to the ones shown in this chapter, can be conducted to investigate the impact on the results.

As a final conclusion, the reader should not focus on the exact results in the experiments, since they are data-specific. The key point is that the results show that the methodology works. Consequently, the methodologies introduced in this Thesis can be used as a guideline when setting up an anomaly detection system for own vehicle data.

Chapter 10

Conclusion

This chapter summarises the Thesis, identifies the main contributions, and discusses benefits and limitations of the proposed approach.

This Thesis addressed the problem of having to cope with huge data volumes resulting from vehicle tests. The aim was to report anomalies in recordings from test drives, where the key point was to be able to detect unmodelled faults. The detected anomalies can point the expert to faults in the underlying vehicle, which can for example be erroneous sensors or actuators, faults in the software, hardware or parameterisation of electronic control units or faults in the bus systems.

The aim was achieved by proposing a semi-autonomous detection system. The proposed classifier SVDDsubseq learns from a training set of error-free recordings, and then autonomously reports deviations in the test set as anomalies. By means of visual data mining techniques, the expert can then analyse the reported anomalies in order to find those that are caused by faults.

10.1 Main contributions

This Thesis made the following main contributions:

- Current shortcomings when recording and analysing test drives were identified. (Section 1.1.2)
- Visual data mining techniques were adapted to make the manual analysis of test drives more efficient (Chapter 4).
- The one-class classifier support vector data description (SVDD) was made applicable to practical problems by proposing a parameter tuning approach (Section 6.3.6).
- SVDDSUBSEQ was proposed, enhancing SVDD to work on multivariate time series data. In combination with the parameter tuning approach, SVDDSUBSEQ can be applied to test drive data without the need for parameter tuning.
- An anomaly detection system for test drive data was proposed (Chapter 8) and the idea was shown to work on real recordings from vehicles (Section 9).

10.2 Applicability of the approach

The most essential question to answer is, whether the presented anomaly detection system is useful in practice, where a great number of test drives is recorded on a day-to-day basis.

The presented results on test drive data show the applicability of the approach. In the experiments, different kinds of injected faults were successfully detected.

Inherent in the idea to solely rely on a one-class training set of historical data and confirmed by the results, the system will yield misclassifications. On the one hand an

expert is required to distinguish between real faults and falsely reported anomalies, on the other hand some faults may remain undetected.

The conclusion is that even if misclassifications occur, which is inevitable for classification systems based on learning from sample data, the detection system is a significant improvement. Based on the reported anomalies the expert can conduct the analysis in a goal-oriented manner in contrast to random inspection. In addition, the anomaly score attached to each reported subsequence, allows the results to be ranked.

However, if the vast majority of reported anomalies are incorrect, the process becomes error-prone, because it is tedious for the expert and he/she might overlook those results that are in fact abnormal. To avoid this, the anomaly detection system should be continuously optimised during its operation mode. Selected classification results confirmed by the expert should be integrated into the knowledge base. SVDD, while essentially being a one-class classifier, can be trained with additional instances from the abnormal class (Tax, 2001).

When more representative data sets are available, the detection system should be trained on an enhanced training set. If the results have improved, the enhanced training set can be integrated. For that reason a history of classification results should be kept.

The effectiveness of the approach was quantified for one experiment, yielding a speedup factor of 16 with respect to the currently used techniques. The speedup factor is expected to be orders of magnitude higher in practice since it increases the less faults are present in the data.

10.3 Classification accuracy

The question arises, which classification accuracy can be considered as valuable. For measuring the accuracy, two essential measures were used in this Thesis: (1) the true negative rate, which is the percentage of anomalies that were detected, and (2)

the precision on the abnormal class, which measures the percentage of the reported anomalies that are indeed abnormal.

Obviously a classifier could be created that detects the vast majority of anomalies by drawing a very tight boundary resulting in a high true negative rate. As a result, a large portion of normal instances would be classified as abnormal as well, consequently the precision is very low.

A trade-off between these two measures has to be found. It is basically the trade-off between the cost of overseeing an anomaly and the cost of investigating a reported anomaly that turns out to be false. If these two cost factors can be quantified, the classifier can be adapted for a given training set to yield the optimal results in terms of costs.

10.4 Scalability

The number of incorporated signals is theoretically not constrained. However, the more signals the data set contains, the more variations are possible. Consequently, a very large data set is required in order to have a representative training set.

A sensible limit on the number of signals cannot be given, since it depends on how closely the signals are related. For that reason, a way of determining whether a given training set is representative by determining the false negative rate for different training sets of growing size was proposed in Section 9.

Regarding the length of the recordings used for training, it is a matter of how fast the training result required. It was shown that the complexity of the training period is quadratic in the size of the training set.

The times required for training and test were measured on a current standard PC. The training period for the data set used in Table 9.6 in Section 9.3.1.4, which contains 8 signals and 63631 seconds (17.5 hours) of recordings, takes approximately 20 hours, so the result is available on the next working day. Testing on the other hand is very

fast. For the given training set, testing recordings of 20000 seconds (5.5 hours) takes only 2 seconds. The training time can be reduced by more powerful hardware. Since testing is very fast and done sequentially, the size of the test set is irrelevant.

10.5 Limitations

The approach works on a group of related signals. It essentially learns the value ranges of individual signals and the dependencies between signals. If the relations between signals are time delayed, the detection system will work for relatively small time delays, since the change is then captured by one subsequence. Bigger time delays can be removed by shifting the time series, if the delays are approximately constant. If time delays are not constant, they cannot be learnt by the system. A solution could be to use more refined ways of pre-processing, e.g. to transform the time series into rules (Hoeppner, 2002) or to apply a grammar to describe relations between intervals (Moerchen, 2006).

A further limitation is, that the detection system will not reliably detect unexpected correlations between signals, i.e. signals that should be unrelated but are erroneously related, e.g. by cross-talk. If the erroneous relation leads to an anomaly in the relation of further signals, it can be detected, otherwise not. Alternatively, those type of faults could be detected by means of similarity measures (Mitsa, 2010).

10.6 Benefits of the approach

The proposed approach offers benefits in many steps of a vehicle's life cycle ranging from the development phase to the after sales service period. During test drives conducted before start of production the vehicle's behaviour on the road is evaluated. Analysing the resulting recordings becomes more efficient using the proposed approach. The analysis will be more thorough with a smaller chance of overseeing abnormal behaviour.

After start of production, sporadic test drives are being conducted with selected vehicles to ensure the vehicles' quality. At that point in time, many recordings with that type of vehicle exist from earlier phases. Therefore the system shall be able to offer good results for this step of the vehicle's life cycle. Even after a vehicle has gone through all of the manufacturer's steps, such a system offers benefits: during the analysis of field data.

Chapter 11

Outlook

This chapter discusses possible enhancements of the approach as well as identifying and proposing further research directions.

This work proposed an anomaly detection system without setting pre-configured constraints for the data and without modelling effort. The results show the applicability of the approach. In order to improve the overall fault detection rate, for known and unknown faults, more domain-specific knowledge could be integrated:

1. If the detection system reports an anomaly that is classified by the expert as being irrelevant or incorrect, this information could be added to the knowledge-base. By means of similarity measures, the reporting of similar occurrences can be suppressed. Equivalently the information which anomalies pointed to faults could be stored for further, similar anomalies.

2. If there are constraints for the data, that are reliably known not to be violated in normal operation mode, these constraints could be integrated as well. This follows the common approach of limiting the search space by pushing constraints

deep into a data mining process (Han and Kamber, 2006). An example could be valid value ranges for signals.

3. Well-known fault patterns could be pre-configured, for the system to reliably detect known faults.

The reported anomalies point the experts to potential errors which in turn may point to faults. Further research could be to identify the cause of the fault. This requires pre-configured knowledge, like information about faults and fault symptoms. While this Thesis presented an approach that does not require modelling, such an enhancement would require massive modelling effort and will most likely be constrained to known faults. An alternative with less modelling effort could be to configure knowledge about the in-vehicle network topology, so the detection system is able to isolate the fault location.

In addition to the offline-detection of anomalies, the proposed classifier SVDDSUBSEQ has the potential to be used for online-diagnosis in ECUs for example to store a diagnostic trouble code in case of an error. The diagnostic capabilities of ECUs are limited by computing and memory constraints. With a support vector machine, classification is very fast and the knowledge base can be compact, if the number of support vectors is kept low. However, for that purpose, higher classification accuracies than achieved for the test drives in this Thesis are required. This is feasible if the classifier is trained for one specific diagnostic task, rather than the detection of any type of faults. Optimally SVDDSUBSEQ could be run in parallel to an alternative, conventional diagnostic algorithm.

Apart from fault detection in test drives, a variety of further tasks in the vehicle development process can benefit from this approach:

- The approach could be used to compare the behaviour of different software releases. So for example abnormal deviations of the timing behaviour between software releases can be uncovered.

- The evaluation of an algorithm's behaviour with respect to varied parameterisation is another field to benefit from this approach.

- During the development of vehicle functions, the algorithms running on ECUs are evaluated and optimised permanently. The tests are typically run on a software in the loop (SiL) or hardware in the loop (HiL) environment. The approach can support the responsible specialist during the analysis of the test results by pointing her/him to anomalies.

In a variety of application domains, the amount of data recorded for later analysis is growing due to cheap and compact hardware and storage devices. Consequently the research field of anomaly detection will continue to gain importance. The approach presented in this Thesis is transferable to other application domains where the data has similar properties. It could be applied to data from other technical systems, for example fault detection in recordings from machinery in the automation industry. Managing computer networks, performance monitoring or intrusion detection are potential applications. In marketing research it could be used to identify changes in consumer behaviour. Further applications could be the monitoring of medical data or the management of power-grids.

List of Figures

List of Tables

Bibliography

Abe, S. (2010). *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*. Springer-Verlag London Ltd., 2 edition.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Antunes, C. M. and Oliveira, A. L. (2001). Temporal data mining: an overview. In *KDD 2001 Workshop on Temporal Data Mining*, pages 1 – 13.

Athanasas, K. and Dear, I. (2004). Validation of complex vehicle systems of prototype vehicles. *IEEE Transactions on Vehicular Technology*, 54.

Auto Service Praxis (2013). Website: Data base for product recalls (accessed 5th August 2013).

Ben-Hur, A. and Weston, J. (2010). *A User's Guide to Support Vector Machines*, volume 609 of *Methods in Molecular Biology*. Humana Press.

Berger, C. and Rumpe, B. (2012). Autonomous Driving-5 Years after the Urban Challenge: The Anticipatory Vehicle as a Cyber-Physical System. In *GI-Jahrestagung*, volume 208 of *LNI*, pages 789–798. GI.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, reprinted edition 2006 edition.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. (2005). Fast kernel classifiers with online and active learning. *Machine Learning Research*, 6:1579–1619.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. In *SIGMOD Conference*, pages 93–104.

Byttner, S., Rögnvaldsson, T., and Svensson, M. (2011). Consensus self-organized models for fault detection (COSMO). *Engineering Applications of Artificial Intelligence*, 24(5):833 – 839.

Chandola, V. (2009). *Anomaly Detection for Symbolic Sequences and Time Series Data*. PhD thesis, Computer Science Department, University of Minnesota.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Cong, F., Hautakangas, H., Nieminen, J., Mazhelis, O., Perttunen, M., Riekki, J., and Ristaniemi, T. (2013). Applying wavelet packet decomposition and one-class support vector machine on vehicle acceleration traces for road anomaly detection (accepted for publication). In *Advances in Neural Networks ISNN 2013*.

Dannenberg, J. and Burgard, J. (2007). Car innovation 2015 - a comprehensive study on innovation in the automotive industry. Technical report, Oliver Wyman Automotive.

de Ridder, D., Tax, D., and Duin, R. P. W. (1998). An experimental comparison of one-class classification methods. In Ter Haar Romeny, B., Epema, D., Tonino, J., and Wolters, A., editors, *Proc. 4th Annual Conference of the Advanced School for Computing and Imaging (ASCI 98)*, pages 213–218. ASCI, ASCI.

de Sa, J. P. M. (2001). *Pattern recognition: concepts, methods, and applications*. Springer-Verlag.

Deng, K., Moore, A., and Nechyba, M. (1997). Learning to Recognize Time Series: Combining ARMA models with Memory-based Learning. In *IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, volume 1, pages 246 – 250.

Denton, T. (2006). *Advanced Automotive Fault Diagnosis*. Elsevier Ltd., 2 edition.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification (2nd Edition)*. Wiley-Interscience.

Endres, E., Muller, C., Shadrin, A., and Tverdyshev, S. (2010). Towards the formal verification of a distributed real-time automotive system. In *Proceedings Second NASA Formal Methods Symposium NFM 2010*, pages 212–217.

Etzold, R. (2011). *So wird's gemacht. Pflegen - warten - reparieren: Renault Twingo von 6/93 bis 12/06*. Delius Klasing, 8 edition.

Fawcett, T. (2004). ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories.

Fayyad, U., Piatetsky-shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54.

Ferreira, M. C. and Levkowitz, H. (2003). From visual data exploration to visual data mining: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394.

Few, S. (2006). Multivariate analysis using parallel coordinates. *Perceptual Edge*.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188.

Friedrich, A., Lindau, J., Heinrich, J., Ebner, A., Haug, R., Schmidt, M., Ertelt, C., Dassler, J., Bäumel, A., and Scheible, K. (2009). Testing and Tuning. *ATZ extra. The New E-Class by Mercedes-Benz*.

Fuerst, S. (2010). System and Software Architectures with AUTOSAR Basic and Application Software. In *Steinbeis Symposium Electronics in Automotive Engineering*.

Furnas, G. W. and Buja, A. (1994). Prosection views: Dimensional inference through sections and projections. *Journal of Computational and Graphical Statistics*, 3:323–385.

Grezemba, A. (2011). *MOST. The automotive multimedia network. 2nd Edition.* Franzis.

Grimm, D. K., Sadekar, V., and Popp, P. (2007). A General Motors Perspective on the Deployment of Vehicle to Vehicle Communications based Active Safety and Driver Assistance Applications. In *13th International Conference on Electronic Systems for Vehicles*.

Hackenberg, U. (2008). Innovative Vehicle Architectures for Future Demands. In *32rd FISITA World Automotive Congress*. FISITA.

Han, J. and Kamber, M. (2006). *Data Mining - Concepts and Techniques*. Morgan Kaufmann Publishers, 2 edition.

Hauskrecht, M., Valko, M., Batal, I., Clermont, G., Visweswaram, S., and Cooper, G. (2010). Conditional outlier detection for clinical alerting. *Annual American Medical Informatics Association Symposium*.

Hodge, V. J. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:2004.

Hoeppner, F. (2002). Learning dependencies in multivariate time series. In *Workshop on Knowledge Discovery in (Spatio-) Temporal Data*, pages 25–31.

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. (2003). A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Hwang, I., Kim, S., Kim, Y., and Seah, C. E. (2010). A Survey of Fault Detection, Isolation, and Reconfiguration Methods. *IEEE Transactions on Control Systems Technology*, 18(3):636–653.

Inselberg, A. (1985). The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91.

Isermann, R. (2006). *Fault-Diagnosis Systems – An Introduction from Fault Detection to Fault Tolerance*. Springer, 1 edition.

ISO 26262-1 (2011). ISO 26262: Road vehicles – Functional safety – Part 1: Vocabulary. Final Draft.

Jautze, M., Bogner, A., Eggendinger, J., Fröhlich, M., Rehdorf, J., Rekewitz, G., and Stumm, A. (2008). An innovative variable damper system for further improvement of ride-comfort and handling. In *32rd FISITA World Automotive Congress*. FISITA.

Jeutter, R. (2008). Test and Validation of Distributed Automotive Systems – Communication Robustness Validation. In *Elektronik-Systeme im Automobil*.

Johnson, M. A. (2009). SVM.NET 1.6.3.

Jones, C. A. (2005). Lecture notes: Math2640 introduction to optimisation 4. Technical report, University of Leeds, School of Mathematics.

Jones, W. D. (2001). Keeping cars from crashing. *Spectrum, IEEE*, 38(9):40–45.

KEEL (2012). Website: KEEL (Knowledge Extraction based on Evolutionary Learning).

Keim, D. A. (1997). Visual techniques for exploring databases.

Keim, D. A. (2001). Visual exploration of large data sets. *Communications of the ACM*, 44:38–44.

Keim, D. A. (2002). Information visualization and data mining. *IEEE Transaction on Visualization and Computer Graphics*, 7.

Keogh, E. and Lin, J. (2005). Hot SAX: Efficiently finding the most unusual time series subsequence. In *5th IEEE International Conference on Data Mining (ICDM 2005)*, pages 226–233.

Keogh, E., Lin, J., Lee, S.-H., and Verle, H. V. (2006). HOT SAX: finding the most unusual time series subsequence: algorithms and applications. *Knowledge and Information Systems*, 11(1):1–27.

Kirchgässner, G. and Wolters, J. (2007). *Introduction to Modern Time Series Analysis*. Springer Verlag.

Koch, M. and Theissler, A. (2007). Mit Tedradis dem Fehler auf der Spur. Effizente Analyse von Fehlern im Fahrzeug zwischen Entwicklung und Serienanlauf. *hanser automotive*, 9:28–30.

Krämer, M., Röhringer, A., Kirchner, W., Vogel, T., and Rochlitzer, J. (2009). Programme Management and Project Control. *ATZ extra. The New E-Class by Mercedes-Benz*.

Krauß, S. (2010). Comprehensive ECU Tests with Fault Simulation. Technical report, Vector Informatik GmbH.

Lamberg, K. (2006). Model-based testing of automotive electronics. *Design, Automation and Test in Europe Conference and Exhibition*, 1:28.

Laurikkala, J., Juhola, M., and Kentala, E. (2000). Informal identification of outliers in medical data. In *14th European Conference on Artificial Intelligence ECAI-2000. Berlin.*

Laxman, S. and Sastry, P. (2006). A survey of temporal data mining. *Sadhana*, 31.

Liebemann, E. K., Meder, K., Schuh, J., and Nenninger, G. (2004). Safety and Performance Enhancement: The Bosch Electronic Stability Control(ESP). *SAE Technical Paper Series.*

Mack, B. and Waske, B. (2011). Optimizing support vector data description by automatically generating outliers. In *EARSeL 7th SIG-Imaging Spectroscopy Workshop, Edinburgh.*

Marscholik, C. and Subke, P. (2008). *Road vehicles – Diagnostic communication.* Hüthig GmbH und Co. KG.

MATLAB (2011). *version 7.13 (R2011b).* The MathWorks Inc., Natick, Massachusetts.

Mayer, E. (2010a). Serial Bus Systems in the Automobile. Part 2. Reliable data exchange in the automobile with CAN. Technical report, Vector Informatik GmbH.

Mayer, E. (2010b). Serial Bus Systems in the Automobile. Part 3. Simple and cost-effective data exchange in the automobile with LIN. Technical report, Vector Informatik GmbH.

Mayer, E. (2010c). Serial Bus Systems in the Automobile. Part 4. FlexRay for data exchange in safety-critical applications. Technical report, Vector Informatik GmbH.

Mayer, E. (2010d). Serial Bus Systems in the Automobile. Part 5. MOST for transmission of multimedia data. Technical report, Vector Informatik GmbH.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill Education (ISE Editions).

Mitsa, T. (2010). *Temporal Data Mining.* Chapman & Hall/CRC.

Moerchen, F. (2006). *Time Series Knowledge Mining*. PhD thesis, Philipps-Universität Marburg.

Moseler, O. and Isermann, R. (2000). Application of model-based fault detection to a brushless dc motor. *IEEE Transactions on industrial electronics*, 47:1015–1020.

Moya, M. M. and Hush, D. R. (1996). Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474.

Mueter, M. and Asaj, N. (2011). Entropy-based anomaly detection for in-vehicle networks. In *Intelligent Vehicles Symposium*, pages 1110–1115. IEEE.

Myers, G. J. (2004). *The Art of Software Testing, Second Edition*. Wiley.

Olszewski, R. T. (2001). *Generalized Feature Extraction for Structural Pattern Recognition in Time Series Data*. PhD thesis, School of Computer Science, Carnegie Mellon University.

Pavlichenko, O. (2011). Adaptation of measured data analysis algorithms for an existing machine learning framework.

Pons, F. S., Fernandez, D. S., Tort, M. S., Garcia, P. G., and Rodriguez, A. B. R. (2010). Data fusion strategies for next generation ADAS: Towards full collision avoidance. In *33rd FISITA World Automotive Congress*. FISITA.

PRTools (2012). Website: PRTools: The Matlab Toolbox for Pattern Recognition.

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Raykov, T. and Marcoulides, G. (2012). *Basic Statistics: An Introduction with R*. Rowman & Littlefield Publishers.

Sander, O., Klimm, A., Becker, J., Becker, J., Kimmeskamp, T., Formann, J., Echtle, K., Weinberger, K., and Bulach, S. (2009). Ensuring reliability and interoperability for intra vehicular communication by formal verification. In *14th International Conference on Electronic Systems for Vehicles*.

Schäuffele, J. and Zurawka, T. (2005). *Automotive Software Engineering: Principles, Processes, Methods, and Tools*. SAE International, 4. edition.

Schlingmann, N. (2008). Diagnostics in the field by CLAAS. In *5th CTI Forum on Automotive Diagnostic Systems*.

Schlinkheider, J. (2010). Elektromobilität – Ruetteln an Grundmanifesten. In *Steinbeis Symposium Electronics in Automotive Engineering*.

Schmidgall, R. (2011). Diagnostic communication. Opportunities and challenges. In *8th CTI Forum on Automotive Diagnostic Systems*.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *In Proceedings of Visual Languages. IEEE Computer Science Press.*, pages 336–343.

Shumway, R. H. and Stoffer, D. S. (2006). *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics. Springer Science+Business Media, 2 edition.

Sommerville, I. (2001). *Software Engineering (6th Edition)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Song, X., Wu, M., Jermaine, C., and Ranka, S. (2007). Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):631–645.

Stein, B., Niggemann, O., and Balzer, H. (2006). Diagnosis in Automotive Applications: A Case Study with the Model Compilation Approach. In *Third Monet Workshop on Model-Based Systems at the ECAI*, pages 34–40.

Suwatthikul, J. (2008). *A Framework and Methods for On-board Network Level Fault Diagnostics in Automobiles*. PhD thesis, School of Engineering, University of Warwick.

Suwatthikul, J., McMurran, R., and Jones, R. (2011). In-vehicle network level fault diagnostics using fuzzy inference systems. *Applied Soft Computing*, 11(4):3709 –

3719.

Svensson, M., Byttner, S., and Rognvaldsson, T. (2008). Self-organizing maps for automatic fault detection in a vehicle cooling system. In *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, volume 3.

Tax, D. and Duin, R. (2001a). Outliers and data descriptions. In *In Proceedings of the Seventh Annual Conference of the Advanced School for Computing and Imaging (ASCI)*.

Tax, D. and Duin, R. (2004). Support vector data description. *Machine Learning*, 54(1):45–66.

Tax, D. M. (2001). *One-class classification. Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology.

Tax, D. M. and Duin, R. P. (1999). Data domain description using support vectors. In *Proceedings of the European Symposium on Artificial Neural Networks*, pages 251–256.

Tax, D. M. and Duin, R. P. (2001b). Uniform object generation for optimizing one-class classifiers. *Journal of Machine Learning Research*, 2:155–173.

Theissler, A. and Dear, I. (2012). Detecting anomalies in recordings from test drives based on a training set of normal instances. In *Proceedings of the IADIS International Conference Intelligent Systems and Agents 2012 and European Conference Data Mining 2012. IADIS Press, Lisbon.*, pages 124–132.

Theissler, A. and Dear, I. (2013a). An anomaly detection approach to detect unexpected faults in recordings from test drives. In *Proceedings of the WASET International Conference on Vehicular Electronics and Safety 2013, Stockholm.*, pages 1144–1151.

Theissler, A. and Dear, I. (2013b). Autonomously determining the parameters for SVDD with RBF kernel from a one-class training set. In *Proceedings of the WASET International Conference on Machine Intelligence 2013, Stockholm.*, pages 1135–1143.

Theissler, A., Palmer, J., Rehborn, H., and Dear, I. (2011). Interactive Anomaly Detection in time series resulting from local traffic measurements. In *Proceedings*

of the IADIS European Conference Data Mining 2011. IADIS Press, Lisbon., pages 147–152.

Theissler, A., Ulmer, D., and Dear, I. (2010). Interactive knowledge discovery in recordings from vehicle tests. In *33rd FISITA World Automotive Congress*. FISITA.

Theodoridis, S. and Koutroumbas, K. (2009). *Pattern Recognition, Fourth Edition*. Academic Press, 4th edition.

Thomas, J. J. and Cook, K. A. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr.

Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S. (2003a). A review of process fault detection and diagnosis. Part II: Qualitative methods and search strategies. *Computers and Chemical Engineering*, 27(3):293–311.

Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S. (2003b). A review of process fault detection and diagnosis. Part III: Process History Based Methods. *Computers and Chemical Engineering*, 27(3):293–311.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S. N. (2003c). A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers and Chemical Engineering*, 27(3):293–311.

von Glasner, C. and Micke, S. (2010). Driver Assistance Functions. Status 2010. In *33rd FISITA World Automotive Congress*. FISITA.

Wang, L., Froehlich, H., Rieck, K., Tsai, C.-T., and Lin, T.-J. (2010). LIBSVM for SVDD and finding the smallest sphere containing all data.

Wattenberg, M. (2001). Sketching a graph to query a time-series database. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 381–382, New York, NY, USA. ACM.

Wegman, E. J. (2003). Visual data mining. Center for Computational Statistics, George Mason University.

Weinmann, M., Elshabrawy, K., and Bäcker, B. (2009). Development of a flexible E/E-System architecture for conventional and electrified powertrains. In *14th In-*

ternational Conference on Electronic Systems for Vehicles.

Wernicke, M. (2010). AUTOSAR on its way to production. Technical report, Vector Informatik GmbH.

Wolff, T. (2009). Audi's E-Architecture: Challenges in the new A8. In *14th International Conference on Electronic Systems for Vehicles*.

Wolfsried, S. (2009). Environmental Friendly, Safe and Comfortable Cars by Modular E/E Systems. In *14th International Conference on Electronic Systems for Vehicles*.

Zhuang, L. and Dai, H. (2006). Parameter optimization of kernel-based one-class classifier on imbalance learning. *Journal of Computers*, 1(7):32–40.